



TRABAJO FIN DE GRADO
GRADO EN INGENIERIA INFORMATICA

Título

Subtítulo

Francisco de Asís Carrasco Conde

Autor

Francisco de Asís Carrasco Conde

Directora

María José Rodríguez Fórtiz



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

—
Granada, Junio de 2025

Título
Subtítulo

Francisco de Asís Carrasco Conde

Palabras clave: *software libre*

Resumen

Same, but in English

Student's name

Keywords: *open source, floss*

Abstract

Yo, **Nombre Apellido1 Apellido2**, alumno de la titulación **TITULACIÓN de la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI XXXXXXXXX, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Nombre Apellido1 Apellido2

Granada a X de mes de 201 .

D. **María José Rodríguez Fórtiz**, Profesora del Departamento de Lenguajes y Sistemas Informáticos

Informo:

Que el presente trabajo, titulado ***Nombre de la App***, ha sido realizado bajo mi supervisión por **Francisco de Asís Carrasco Conde**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a Junio de 2025.

El/la director(a)/es:

María José Rodríguez Fórtiz

Agradecimientos

Índice general

1. Introducción	21
1.1. Justificación	21
1.2. Objetivos	22
1.3. Estructura de la Memoria	23
2. Estado del arte	25
2.1. Dominio del problema	25
2.2. Aplicaciones similares	26
2.3. Metodologías potenciales a aplicar	30
2.4. Tecnologías potenciales para usar	31
2.4.1. Base de datos	31
2.4.2. Backend	33
2.4.3. Frontend	35
3. Propuesta	37
3.1. Descripción detallada	37
3.2. Elección de tecnologías	39
3.3. Backend	39
3.4. Frontend	39
3.5. Base de datos	39
3.6. Diagrama de arquitectura	40
3.7. Metodología utilizada	40
3.8. Temporización	40
3.9. Seguimiento del desarrollo	41
3.9.1. Iteración 0	41

3.9.2.	Iteración 1	86
3.9.3.	Iteración 2	95
3.9.4.	Iteración 3	115
3.9.5.	Iteración 4	129
3.9.6.	Iteración 5	147

Índice de figuras

2.1. Fitbit	27
2.2. Apple fitness	28
2.3. Strava	29
2.4. Pulsera whoop	30
2.5. Tabla comparativa tipos de bases de datos	33
2.6. Tabla comparativa frameworks para el backend	34
2.7. Tabla comparativa frameworks para el frontend	36
3.1. Tabla comparativa	38
3.2. Tabla comparativa	40
3.3. Pagina inicial	43
3.4. Inicio de sesion	44
3.5. Crear cuenta 1	45
3.6. Crear cuenta 2	46
3.7. Menu principal	47
3.8. Lista ejercicios	48
3.9. Lista ejercicios filtrada	49
3.10. Datos ejercicio	50
3.11. Pop up graficar segun tiempo	51
3.12. Pop up establecer meta	51
3.13. Pop up crear ejercicio	52
3.14. Lista rutinas	53
3.15. Lista rutinas filtradas	54
3.16. Datos rutina modificable	55
3.17. Datos rutina no modificable	55
3.18. Modificar ejericios rutina	56
3.19. Cambiar orden ejercicios rutina	57
3.20. Ejercicios rutina descargada	58

3.21. Pop up crear rutinas	59
3.22. Opciones de perfil usuario	60
3.23. Pop up de confirmacion	61
3.24. Pop up de confirmacion resumir datos	61
3.25. Buscar usuario	62
3.26. Buscar usuario filtrado	63
3.27. Rutinas de un usuario	64
3.28. Widget descargar rutina	65
3.29. Widget rutina descargada	65
3.30. Buscar rutinas por su nombre filtrada	66
3.31. Entrenamiento de un día determinado	67
3.32. Descanso en un día determinado	68
3.33. Frame 29.png	69
3.34. Lista ejercicios del entrenamiento actual	70
3.35. Primera serie flexiones	71
3.36. Realizando flexiones	72
3.37. Fin de la serie(Descanso no completado)	73
3.38. Fin de la serie(Descanso completado)	74
3.39. Acabar ejercicio o añadir serie	75
3.40. Realizando ejercicio midiendo tiempo	76
3.41. Fin entrenamiento	77
3.42. Metas cumplidas	78
3.43. Datos entrenamiento terminado	79
3.44. Chat con IA	80
3.45. Datos detallados entrenamiento terminado	81
3.46. Pop up datos de un ejercicio en entrenamiento	82
3.47. Backlog prioridades MoSCoW	84
3.48. Backlog prioridades MoSCoW	85
3.49. Lista ejercicios	87
3.50. Crear ejercicios	89
3.51. Modificar descripcion ejercicios	90
3.52. Nueva meta	92
3.53. PopUp Confirmacion	93
3.54. BD local iteracion 1	94
3.55. Pantalla nueva	95
3.56. Pantalla antigua	95
3.57. Comparación entre la pantalla nueva y la anterior	95
3.58. Menu Principal semifuncional	97
3.59. Log In / Sing In	99
3.60. Sing In pantalla 1	100
3.61. Sing In pantalla 2	101
3.62. Log In	103
3.63. Opciones Usuario	104
3.64. Lista Rutinas	106

3.65. Datos Rutina	108
3.66. Crear Rutina	109
3.67. Lista Ejercicios Rutina	110
3.68. Modificar Rutina	111
3.69. Lista añadir ejercicio a rutina	112
3.70. BD local iteracion 2	114
3.71. BD backend iteracion 2	115
3.72. Peso Objetivo	116
3.73. Elegir lista compartida por el usuario o locales	118
3.74. Rutinas compartidas por el usuario	119
3.75. Datos rutinas compartidas por el usuario	120
3.76. Buscar rutinas o usuario	122
3.77. Buscar rutinas por su nombre	123
3.78. Buscar usuarios	124
3.79. Datos rutinas para descargar	126
3.80. BD backend iteracion 3	128
3.81. Calendario	131
3.82. Calendario con eventos	132
3.83. Descanso	133
3.84. Entrenar	134
3.85. Lista inicial de ejercicios	136
3.86. Realizando serie	137
3.87. Guardando marcas	138
3.88. Pantalla enter series	139
3.89. Flujo entrenamiento	140
3.90. Meta superada	142
3.91. Registrar peso por día	144
3.92. BD local iteración 4	146
3.93. BD backend iteración 4	147

Índice de tablas

Capítulo 1

Introducción

1.1. Justificación

En este TFG se va a desarrollar una aplicación que ayude a las personas a realizar ejercicio físico de forma controlada y a monitorizar las metas que se van alcanzando. Primero de todo, ¿qué lleva a hacer este TFG? Para empezar, es verdad que hay muchas apps en los repositorios que están pensadas para ser usadas por usuarios de gimnasio, para hacer más fácil el seguimiento y evolución. Además la gente que suele hacer deporte de manera más informal o por hobby no suele tener los conocimientos o dispositivos para hacer mediciones de calidad e interpretarlo correctamente, por lo que necesitarían una aplicación más sencilla tipo agenda para planificar y monitorizar sus ejercicios.

Otra problemática que encontramos en las apps existentes suele ser la falta de accesibilidad, pensando en usuarios con algún tipo de discapacidad que quieran realizar ejercicios. Por ejemplo, encontramos que los scrolls abundan, hay eventos no controlados por el usuario, colores confusos para daltónicos, a veces hay ausencias de iconos asociados a botones y listas, y un amplio etcétera.

Otras carencias que se observan es que no suelen incluir la funcionalidad de ver entrenamientos recomendados por otros. La gente suele buscarlos en redes sociales o videos que se encuentran en la red, y muchas veces en estos videos se ponen a dar rodeos para rascar más tiempo, así es como el usuario pierde tiempo para que a lo mejor no sea el entrenamiento que el andaba buscando. Lo ideal sería que los ejercicios estuvieran bien organizados en rutinas y que estas pudieran compartirse en la aplicación.

Me veo en la necesidad de hacer esta app para dotar de una herramienta simplificada, fácil de manejar, que permita compartir información entre usuarios

de forma eficaz y accesible. Simplificada porque la información que manejo se trata de una forma fácil e intuitiva, con un poco de experiencia en el deporte se sabe qué significa cada parámetro a medir en un ejercicio. Fácil de manejar, dado que el usuario solo tendrá que, por ejemplo, anotar el número de veces que levanta una pesa y anotarlo en la app. Compartir información eficazmente, porque en una sección de la app habrá una parte formato red social, que permitirá tanto buscar usuarios que comparten entrenamientos, como los propios entrenamientos en sí, acompañados de su descripción en la que el creador da una breve información acerca del entrenamiento. Accesible, porque se seguirán guías de diseño para facilitar el uso por usuarios de diversas capacidades.

Una vez explicado esto, ¿cómo se le dará soporte a los usuarios? Creando una app que le ayude de la siguiente forma: Ayudándole a planificar o usar rutinas de entrenamientos compuestas por series de ejercicios; Facilitándole la monitorización y supervisión de su realización, al poder introducir metas a alcanzar en parámetros, que pueden ser repeticiones/peso/distancia/tiempo, y que se pueden medir de forma independiente o al mismo tiempo. También guardando la cantidad de series de un ejercicio realizadas en un entrenamiento con sus respectivos parámetros, y contabilizándole al usuario el tiempo descansado, para facilitar su correcto entrenamiento. Además, si el usuario lo solicita se le enseñará un resumen de su rendimiento en cada ejercicio con respecto a la fecha en la que se realizó el mismo, para que vea su evolución respecto al tiempo.

También la app ayudará a hacer un control de las metas que se proponga el usuario, es decir, si el usuario se propone bajar de peso o aumentar su rendimiento en x ejercicio, la app le recordará las metas que se autopropuso y le avisará cuando las cumpla.

1.2. Objetivos

Quiero que la app sea un historial interactivo del deporte realizado por el usuario.

Los objetivos de este trabajo de fin de grado son:

- Analizar algunas apps del mercado, así como sus características, qué ofrecen, su costo para el usuario y las valoraciones de los usuarios finales, para aclarar qué es lo que buscan los usuarios en este tipo de apps y considerar esas necesidades en el software a desarrollar.
- Investigar sobre qué herramientas usar para la implementación de la base de datos, backend, frontend, y técnicas y metodologías para dar un desarrollo de calidad al software y para ayudar a la sostenibilidad del sistema en el tiempo.

- Conocer y aprender a usar herramientas actuales y punteras para el desarrollo de aplicaciones móviles.
- Desarrollar una aplicación multiplataforma que cubra todos los requisitos necesarios para dar soporte a la planificación y realización de entrenamientos y monitorización de rutinas de ejercicio físico.
- Tratar de obtener un software lo más accesible posible.

1.3. Estructura de la Memoria

El 1º capítulo(**Introducción**) es una descripción sobre lo que se va a abordar y qué ideas iniciales se tienen acerca del software que se va a desarrollar y algunos aspectos importantes del gremio en el que se usaría.

El 2º capítulo(**Estado del arte**), es un análisis de cómo está el ámbito sobre el que se va a desarrollar este trabajo, es decir, apps del mercado(sus prestaciones y funciones más importantes) y frameworks que se podrían emplear para el desarrollo, así como sus ventajas y desventajas.

El 3º capítulo,

El 4º capítulo(**Conclusiones y trabajos futuros**),

Capítulo 2

Estado del arte

2.1. Dominio del problema

Para comenzar, es necesario hablar un poco sobre conceptos generales en el mundo del deporte, definiendo algunos conceptos. Un ejercicio es la repetición de un movimiento varias veces para estimular uno o varios músculos. Las rutinas, entendámoslos como la colección de distintos ejercicios destinados a entrenar una parte o varias del cuerpo, a las rutinas también se les llama entrenamientos, pero en la app se referirá a rutinas como la anterior definición y los entrenamientos serán el hecho de haber realizado una rutina. Ejemplo, el entrenamiento del día 6 es hacer la rutina de flexiones. Una serie es cuando dentro de un ejercicio repetimos ese movimiento un número determinado de veces. Entre series se hacen descansos. Una repetición, es la realización del movimiento de ese ejercicio en una serie, es decir, si por ejemplo estoy haciendo flexiones, hago 12, descanso, hago 10, descanso, hago 9 y ya no hago más flexiones, dentro de mi entrenamiento se vería así:

Rutina de entrenamiento 1:

- Ejercicio 1
 - Serie 1: X repeticiones
 - Descanso
 - Serie 2: X repeticiones
- Flexiones
 - Serie 1: 12 repeticiones

- Descanso
 - Serie 2: 10 repeticiones
 - Descanso
 - Serie 3: 9 repeticiones
- Ejercicio 3
- Serie 1: X repeticiones
 - Descanso
 - Serie 2: X repeticiones

Las metas son objetivos que se pone el usuario en un ejercicio, por ejemplo, si un usuario desea llegar a hacer 10 flexiones mínimo en una serie la proxima vez que entrene, es que se ha puesto una meta de 10 flexiones.

El compartir una rutina es publicar los detalles de una rutina para que otra persona la pueda hacer en sus entrenamientos.

En este gremio del deporte es muy importante la planificación, ya que si se entrena muy de seguido un músculo te puedes llegar a lesionar o hacer que los entrenamientos sean inútiles, lo que se conoce como sobrentrenamiento. Por su contraparte, es importante no dejar de entrenar todas las zonas del cuerpo, ya que te puede lastrar una musculatura debil para otros tipos de entrenamientos. También es importante planificar los días de descansos, para evitar entrenamientos con el cuerpo fatigado.

Ya dentro de los entrenamientos también es importante respetar los descansos. Dado que si se descansa menos tiempo del debido puede provocar lesiones de muchos tipos, incluido neuronales dado al estrés que sufre el cerebro durante el deporte.

Aunque hagamos correctamente lo mencionado en los 2 párrafos anteriores, puede que el deportista no vea los frutos de sus entrenamientos porque o bien no se acuerda de su rendimiento de hace 1 mes y no sabe si es el mismo o no, o simplemente es un sentimiento placebo. Por ello sería útil tener un historial de entrenamiento de fácil acceso o incluso gráficas, para ver si el deportista a mejorado su rendimiento o no.

2.2. Aplicaciones similares

En esta sección describiremos algunas de las aplicaciones que existen similares a la propuesta. Las que permiten mediciones de constantes relacionadas durante el entrenamiento físico suelen ser de pago.

Ejemplos de aplicaciones con precios mensuales:

- **Fitbit Premium: €9.99**, esta app ofrece una interfaz para medir constantes usando un smartwatch con sensor cardíaco (ritmo cardíaco, calorías quemadas, pasos, ect). En lo que respecta que es el entrenamiento, solo ofrece una serie de entrenamiento prefijados, además no permite guardar resultados.

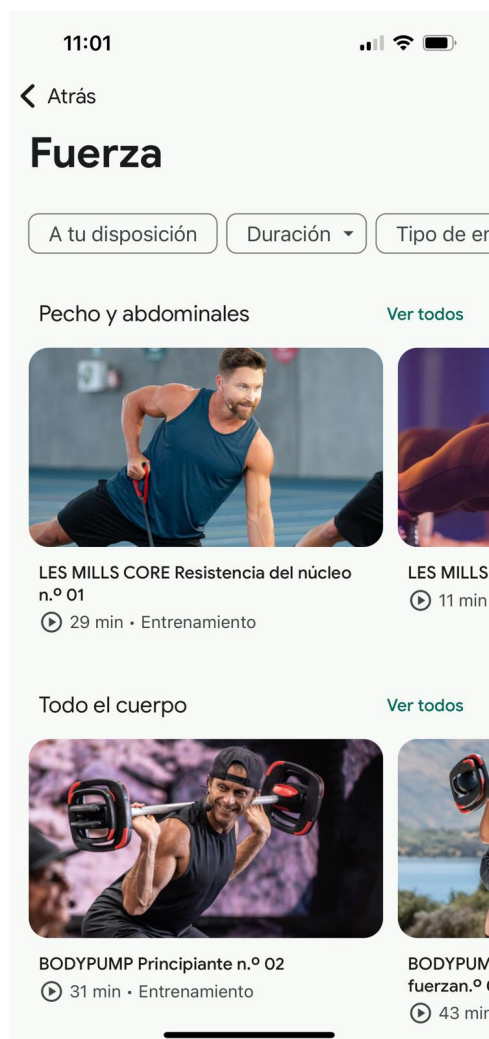


Figura 2.1: Fitbit

- **Apple Fitness+: €9.99**, en su versión gratuita nos permite guardar todas las variables permitidas con un smartwatch, las registra y va aconsejando, si detecta que tienes estrés salta un aviso, poco movimiento, pulsaciones anormales, etc. En la versión de pago añade los entrenamientos, pero vuelve a la misma problemática, son entrenamientos prefijados y no permite

guardar tu rendimiento.



Figura 2.2: Apple fitness

- **Strava Premium: €5.99**, es una app más enfocada a running, pero permite medir variables relacionadas con este tipo de entrenamientos, kilómetros recorridos, ritmo, ruta recorrida, tiempo, etc. Es muy completa solo para ese tipo de entrenamientos.

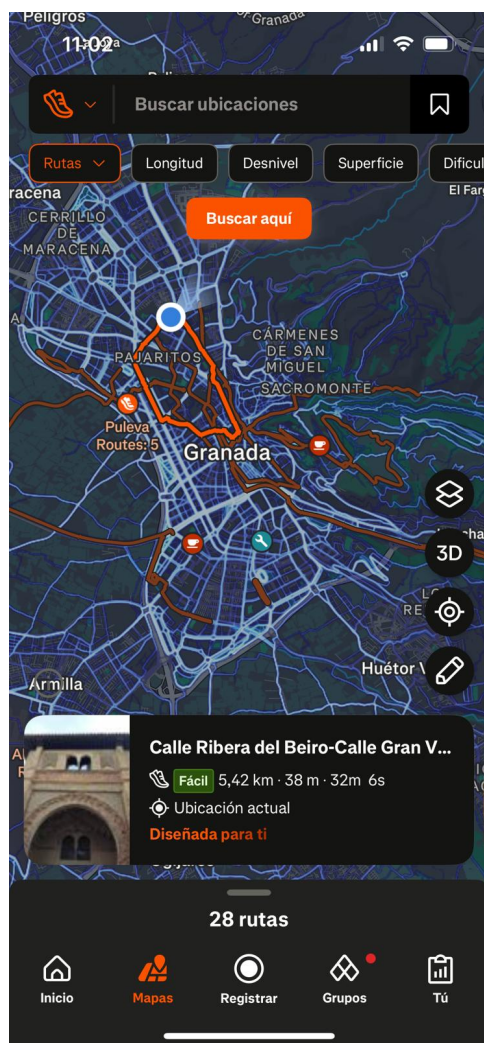


Figura 2.3: Strava

- **Whoop: €28**, de las aplicaciones esta es de lejos la más completa, permite medir, estrés, cansancio del usuario, calidad del entrenamiento, calidad de sueño, recuperación, edad de tu cuerpo según tus constantes, IA especializada para chatear y recibir consejos, etc. Sin embargo, es la que tiene la mensualidad más cara y solo funciona con un reloj de su marca, es decir, el usuario está obligado a comprar ese reloj si quiere usar dicha app, el reloj en concreto vale 200€ mínimo.



Figura 2.4: Pulsera whoop

Es verdad que algunas tienen versión gratuita, pero no ofrecen la totalidad de sus funcionalidades, de hecho, estas versiones suelen ser extremadamente limitadas. Otras aplicaciones como Garmin Connect, sí que son gratuitas, pero te obligan a usar un smartwatch de la marca Garmin, cuyo precio no baja de los 300€.

2.3. Metodologías potenciales a aplicar

Existen muchas formas de desarrollar software con sus respectivas ventajas, desventajas y forma de trabajar. En este caso, no voy a investigar sobre metodologías que usen la idea de prototipos (software que puede ser probado por el cliente durante el desarrollo):

- **Cascada:** Es una forma de desarrollar de las más clásicas y lineales. Se hace una secuencia de fases de desarrollo, las cuales suelen ser diseño requisitos, diseño, implementación, pruebas, implementación y mantenimiento. Es una metodología fácil de entender y aplicar, eficaz para plazos de entrega estrictos. Sin embargo, no es flexible, si se encuentran fallos en la planificación o alguna de las fases, no hay mucho tiempo de reacción.
- **Desarrollo Ágil:** En lugar de una secuencia rígida de fases, se promueve la flexibilidad, ya que se combina con reuniones con el cliente para obtener feedback del software funcional desarrollado entregado en dichas reuniones (no es un prototipo ya que solo es una demostración, no se le entrega para su uso durante el desarrollo), permitiendo corregir errores de planificación o en el propio software durante el desarrollo, sin perder mucho tiempo. Es necesario un buen feedback del cliente.
- **Lean Software Development:** Se basa en la idea de descartar todo aquello que no aporte valor para el cliente y quitarle importancia al de-

sarrollo para ganar calidad y sostenibilidad en el tiempo al software. Esto se refleja en que primero se hace una investigación a fondo de todas las herramientas posibles a usar, su consecuente aprendizaje y en lo más tardío de todo esta la elección de que herramientas usar. Una vez hecha la elección, ya se puede empezar con el desarrollo.

- **V Model:** Es como el método cascada, pero antes de pasar a la siguiente fase se realizan las pruebas de la actual y no se procede hasta que estas estén válidas. Sigue teniendo las mismas desventajas que el modelo de cascada, pero es ideal para proyectos en los que es necesario una alta calidad.
- **Modelo en espiral:** En cada fase se realiza planificación, diseño, desarrollo y evaluación de riesgos. Es muy flexible y esta analizando constantemente los riesgos, pero se requiere cierta experiencia para su correcto empleo.

2.4. Tecnologías potenciales para usar

Para hablar de las tecnologías a emplear, lo voy a separar en las 3 partes importantes en un desarrollo como el que se va a abordar: la base de datos, el backend y el frontend.

2.4.1. Base de datos

En este punto vamos a hablar de las ventajas y desventajas entre usar BDs relacionales y no relacionales, así como las herramientas a emplear en cada una para desarrollar estos servicios.

Relacionales: Las bases de datos relacionales o SQL, como bien sabemos nos permiten guardar datos de una manera bien definida y estructurada, permitiendo asegurar la integridad de la información almacenada. Son muy buena opción, si se prefiere una escalabilidad vertical, es decir, aumentar la capacidad de procesamiento del servidor que va a albergar la BD. Algunas de los SGBDS existentes en la actualidad son:

- MySQL
- MariaDB
- Oracle Database
- SQL server(Microsoft)

- PostgreSQL

No relacionales: También llamadas NoSQL, sirven para trabajar con estructuras de datos semidefinidas o no definidas. Esto quiere decir que no siguen un esquema rígido, lo cuál puede complicar consultas complejas, pero permiten una gran escalabilidad horizontal, permiten añadir más servidores para que operen con la misma base de datos, por tanto aumentar el número de peticiones que puede atender. Cabe decir que dentro de este tipo de bases de datos existen varios subtipos, cada uno especializado en una cualidad:

- Documentales(p.e. MongoDB): guardan los datos en un JSON
- Columnares(p.e. Cassandra): los datos no se guardan en filas, sino en columnas ,ideal para tratar grandes volúmenes de datos por columnas
- Clave-Valor(p.e. DynamoDB): acceso rápido a los datos
- Graficas(p.e. Amazon Neptune): para tratar relaciones complejas

	SQL	NoSQL
Estructura de datos	Datos estructurados con esquema rígido (tablas, filas, columnas)	Datos semi-estructurados o no estructurados (JSON, columnas, clave-valor)
Modelo de datos	Relacional	Varios modelos: documental, columnares, clave-valor, gráficas
Integridad de datos	Alta	Variable
Escalabilidad	Vertical (mejorar hardware de un solo servidor)	Horizontal (añadir más servidores para distribuir carga)
Consultas complejas	Soportadas y optimizadas	Menos eficientes o más complicadas para consultas complejas
Casos de uso típicos	Sistemas con datos estructurados, transacciones, bancos, ERP	Big data, datos no estructurados, aplicaciones web, IoT, análisis en tiempo real
Flexibilidad en esquema	Baja	Alta
Rendimiento en volumen alto	Puede verse limitado por la escalabilidad vertical	Muy buena para volúmenes grandes y distribuidos

Figura 2.5: Tabla comparativa tipos de bases de datos

2.4.2. Backend

En el desarrollo del backend existen varios frameworks que nos van a permitir un desarrollo rápido, eficaz y de calidad. Nuestro backend se va encargar principalmente de recibir y realizar peticiones de los clientes y/o consultas a la BD. Para ello se han investigado las siguientes herramientas:

- Express.js(p.e. Node.js): muy escalable y alto rendimiento

- Django(p.e. Python): ideal para sistemas a gran escala
- Ruby on rails: el mejor en velocidad de desarrollo

Existen más frameworks, pero he decidido investigar solo sobre aquellos que me permitan un desarrollo más ágil, el resto de herramientas como Spring Boot(Java) o Laravel(PHP), también son buenas herramientas, pero tengo más familiaridad con algunas de las herramientas anteriores.

	Express.js (Node.js)	Django (Python)	Ruby on Rails
Lenguaje base	JavaScript	Python	Ruby
Rendimiento	Alto, muy escalable	Muy bueno, adecuado para gran escala	Bueno, optimizado para desarrollo rápido
Velocidad de desarrollo	Rápida, modular y flexible	Rápida, con muchas funcionalidades integradas	Muy rápida, con convenciones que agilizan el desarrollo
Facilidad de aprendizaje	Fácil para desarrolladores JS	Moderada, requiere conocer Python	Moderada, requiere conocer Ruby
Comunidad y soporte	Amplia comunidad, mucha documentación	Comunidad sólida y madura	Comunidad activa y enfocada en productividad
Casos de uso típicos	APIs REST, aplicaciones en tiempo real, microservicios	Aplicaciones web completas, proyectos grandes	Proyectos con desarrollo ágil, startups, MVPs
Flexibilidad	Muy alta, minimalista	Completo y con muchas herramientas incluidas	Alta, con convenciones fuertes para facilitar buenas prácticas
Herramientas integradas	Básico, depende de módulos externos	Incluye ORM, sistema de autenticación, administración	Incluye ORM, sistema de rutas, validaciones integradas

Figura 2.6: Tabla comparativa frameworks para el backend

2.4.3. Frontend

Como se dijo en la introducción, uno de los objetivos es que el software sea multiplataforma, así que se investigará sobre frameworks que me permitan esa capacidad:

- Flutter(lenguaje Dart): de las manos de Google, esta herramienta permite un desarrollo rápido ya que nos permite el hot reload y no tener que recompilar la app cada vez que haya un cambio. También es muy personalizable en lo que respecta a la UI.
- React Native(lenguaje JavaScript): es un framework que extiende React, permitiendo hacer sentir aplicaciones en js como si fueran nativas.
- Xamarín(Lenguajes C#): de Microsoft, esta herramienta es ideal para desarrolladores familiarizados con C# y .NET, también es idóneo para el alto rendimiento y acceso a HW nativo.

	Flutter (Dart)	React Native (JavaScript)	Xamarin (C#)
Lenguaje base	Dart	JavaScript (extensión de React)	C# (.NET)
Velocidad de desarrollo	Alta, con hot reload para ver cambios al instante	Alta, con hot reload	Moderada, requiere compilación
Experiencia UI	Muy personalizable, widgets propios para UI nativa	UI basada en componentes nativos	Acceso a controles nativos para UI
Rendimiento	Muy cercano a nativo	Bueno, aunque puede depender de puentes JS nativos	Muy cercano a nativo, alto rendimiento
Facilidad de aprendizaje	Requiere conocer Dart	Fácil si se conoce JavaScript y React	Más fácil para desarrolladores C#.NET
Acceso a hardware nativo	Sí, mediante plugins	Sí, mediante módulos nativos	Sí, acceso directo al hardware nativo
Comunidad y soporte	Creciente, fuerte respaldo de Google	Amplia comunidad y ecosistema React	Comunidad sólida dentro del ecosistema Microsoft
Casos de uso típicos	Aplicaciones móviles con UI personalizada y multiplataforma	Apps móviles multiplataforma con rapidez y flexibilidad	Apps multiplataforma con integración profunda en Windows y otras plataformas

Figura 2.7: Tabla comparativa frameworks para el frontend

Capítulo 3

Propuesta

3.1. Descripción detallada

La app presentada daría la mayoría de funcionalidades de pago de una manera gratuita y aporta su funcionalidad de medir de forma personalizada el rendimiento del deportista. También se añade una IA, una funcionalidad que no vista en muchas aplicaciones del sector y si existen, son de pago. La función de la inteligencia artificial sería la de aconsejar al usuario cuando este lo necesite, sobre su rendimiento y/o parámetros medidos durante su entrenamiento. Para clarificar las diferencias con el resto de apps del mercado:

X	Registra datos alimenticios	Mide rendimiento de un entrenamie	Puedes hacer tus propios entrenamie	Puedes compartir entrenamientos/conten	Es necesario usar un smartwatch	Tienes una IA consultiva	Requiere pagar una subscripción	Permite controlar el peso del usuario	Puede ser usada offline	Establece metas para tus entrenamie
Strava	No	Solo de running	Solo de running	Solo running	No	No	Existe una versión gratuita limitada	No	No	Solo en los de runniing
Apple Fitness	Si	No	No	No	No	No	Existe una versión gratuita limitada	Si	Solo la parte gratuita	No
Fitbit	SI	No	No	No	No	No	No	Si	Solo la parte gratuita	No
Whoop	No	Si	No	No	Si	Si	Si	Si	Si	No
App presentada	No	Si	Si	Si	No	Si	No	Si	Casi en su plenitud	Si

Figura 3.1: Tabla comparativa

3.2. Elección de tecnologías

3.3. Backend

Se usará Express.js(Node.js), la principal razón que se tiene para optar por esta tecnología, es la gran familiaridad que se tiene con el entorno de JavaScript, se conocen las tecnologías y la forma de implementación, su gran flexibilidad para implementar muchas funcionalidades diversas también fué una buena baza para escoger esta herramienta. Django fué la primera descartada, porque como bien dice en la tabla de la sección anterior(fig. 2.6) está más enfocado a proyectos de gran envergadura. Ruby on Rails fue la que más dudas planteó, como se dijo anteriormente es muy buena opción para las metodologías ágiles y como se dirá más adelante, se optará por una metodología ágil para el desarrollo. Pero al final se descartó por el desconocimiento de la herramienta.

3.4. Frontend

Se ha decidido usar Flutter, dado que Xamarin es buena herramienta, pero está más enfocada a un desarrollo en un ambiente Microsoft. React Native también es una buena herramienta, fue principalmente la que más dudas sembró, ya que como se dijo previamente se usará un backend basado en Express.js, esto entonces permitiría unificar entornos lo cual favorecería al desarrollo. pero se descartó por lo personalizable que son las IUs desarrolladas en Flutter respecto a esta tecnología.

3.5. Base de datos

Para las bases de datos, tanto para el backend como para la local de la app se usará una SQL, dado a la familiaridad que se dispone con esta tecnología y la necesidad de unificar el tipo de tecnología del servidor y de la app. Esto anterior se debe a que como bien se ha dicho se usará Flutter como entorno frontend y la herramienta que se facilita para el almacenamiento local en este es SQLite, basada en un modelo relacional.

Los modelos NoSQL, se descartaron principalmente porque las relaciones entre elementos suelen ser más complejas y en este proyecto pueden enrevesarse fácilmente.

3.6. Diagrama de arquitectura

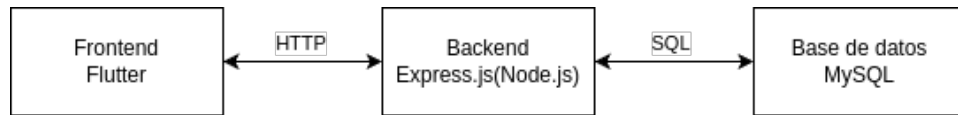


Figura 3.2: Tabla comparativa

3.7. Metodología utilizada

Para el desarrollo de esta app, se usará una metodología *ágil* tipo **SCRUM**. Se decidió usar esta porque permite corregir fallos en la velocidad de diseño y/o planificación de forma eficiente y sin dañar el producto final. Esta metodología permite también entregar desarrollos funcionales a la tutora e ir validando las funcionalidades poco a poco mediante reuniones.

Las reuniones con la tutora serán las *sprint reviews*.

3.8. Temporización

La temporización se realizó el día 25 de marzo de 2025. La entrega del producto (este TFG) está prevista para el 16 de junio de 2025, es decir, 83 días, o lo que es lo mismo, casi 12 semanas. Si un sprint dura 2 semanas, habrá 6 sprints hasta la entrega final.

La iteración 0 se dedicará al diseño de pantallas y al repaso de las funcionalidades de la app, para concretar las historias de usuario y sus prioridades. La idea inicial es dividir la app en varios módulos y centrar cada sprint en cada uno de ellos:

1. Ejercicios
2. Rutinas, usuarios y sesión
3. Descargar y compartir rutinas
4. Flujo de entrenamiento
5. Inteligencia Artificial (IA)
6. Smartwatch y tratamiento de datos

3.9. Seguimiento del desarrollo

3.9.1. Iteración 0

En esta primera iteración me centré en realizar los diseños de la app, considerando su accesibilidad. También concreté el *product backlog*, compuesto por 41 historias de usuario. En las historias no menciono ningún actor puesto que solo existe el usuario que realiza los ejercicios y el es protagonista en todas ellas, no hay ningún moderador.

A pesar de trabajar con SCRUM se realizarán todos los diseños de interfaces de usuario en este primer sprint para concretar con la tutora los requisitos principales de la aplicación a desarrollar en este TFG. En nuestro caso los diseños de las interfaces se usarán como herramientas de captura y especificación de requisitos. En cada iteración se podrán revisar si hay cambios.

La lista inicial de historias de usuario es la siguiente, y algunas incluyen tareas secundarias:

SCRUM-1 Registrar peso por día

SCRUM-2 Establecer peso objetivo

SCRUM-3 Insertar/Borrar/Modificar ejercicio de la lista de ejercicios

SCRUM-4 Buscar rutina en la lista del usuario

SCRUM-5 Insertar/Borrar/Modificar rutina

SCRUM-6 Hacer gráfica en base a las marcas obtenidas

SCRUM-7 Revisar datos para ver si el descanso es necesario

SCRUM-8 Enseñar datos de una rutina a descargar

SCRUM-9 Compartir mi rutina

SCRUM-10 Valorar el entrenamiento en base a la marca actual y la meta del usuario

SCRUM-11 Monitorizar pulso en tiempo real

SCRUM-12 Medir pulso en reposo y compararlo con datos de ejercicios

SCRUM-13 Avisar de anomalías en el pulso de forma suave

SCRUM-14 Obtener calorías quemadas

SCRUM-15 Comprobar el equilibrio nervioso del usuario

SCRUM-16 Realizar el flujo del entrenamiento

SCRUM-17 Conectar con la IA para iniciar diálogo

SCRUM-18 Crear/Borrar usuario

SCRUM-19 Resumir datos

SCRUM-20 Iniciar/Cerrar sesión

SCRUM-21 Medir SpO2

SCRUM-22 Interpretar constantes

SCRUM-23 Buscar ejercicios en la lista de ejercicios

SCRUM-24 Pop up de confirmación

SCRUM-25 Implementar menú principal

SCRUM-26 Detectar metas cumplidas en los ejercicios después del entrenamiento

SCRUM-27 Buscar rutinas para descargar

SCRUM-28 Implementar calendario

A continuación se muestran los diseños creados en esta iteración:

Inicio de sesión y crear usuario

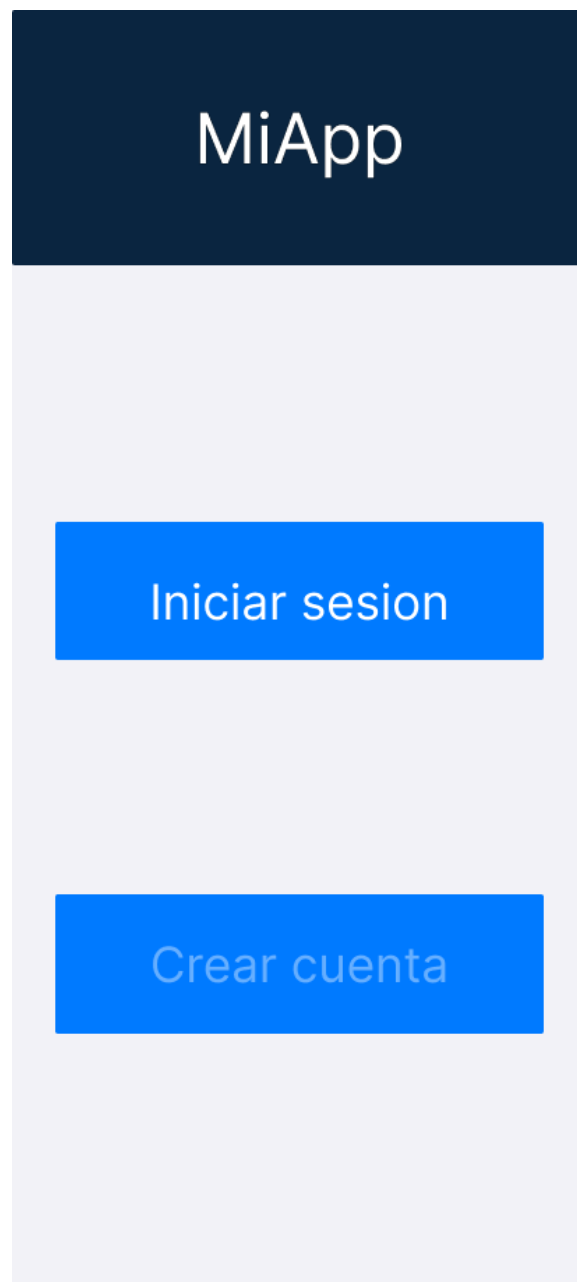
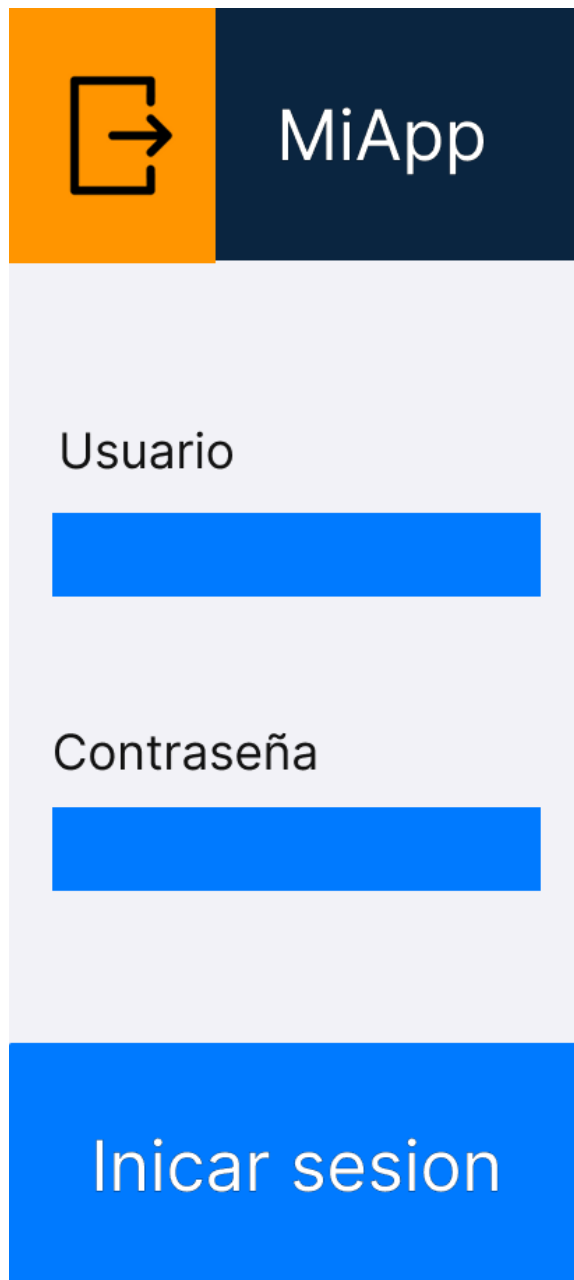



Figura 3.3: Pagina inicial



The image shows a mobile application login screen. At the top, there is a header bar with an orange square on the left containing a white icon of a document with an arrow pointing right, and a dark blue square on the right containing the text "MiApp" in white. Below the header, the background is light gray. There are two text input fields: the first is labeled "Usuario" and the second is labeled "Contraseña". Both input fields are represented by solid blue rectangles. At the bottom of the screen, there is a large blue button with the text "Inicar sesion" in white.

Figura 3.4: Inicio de sesion



MiApp

Correo

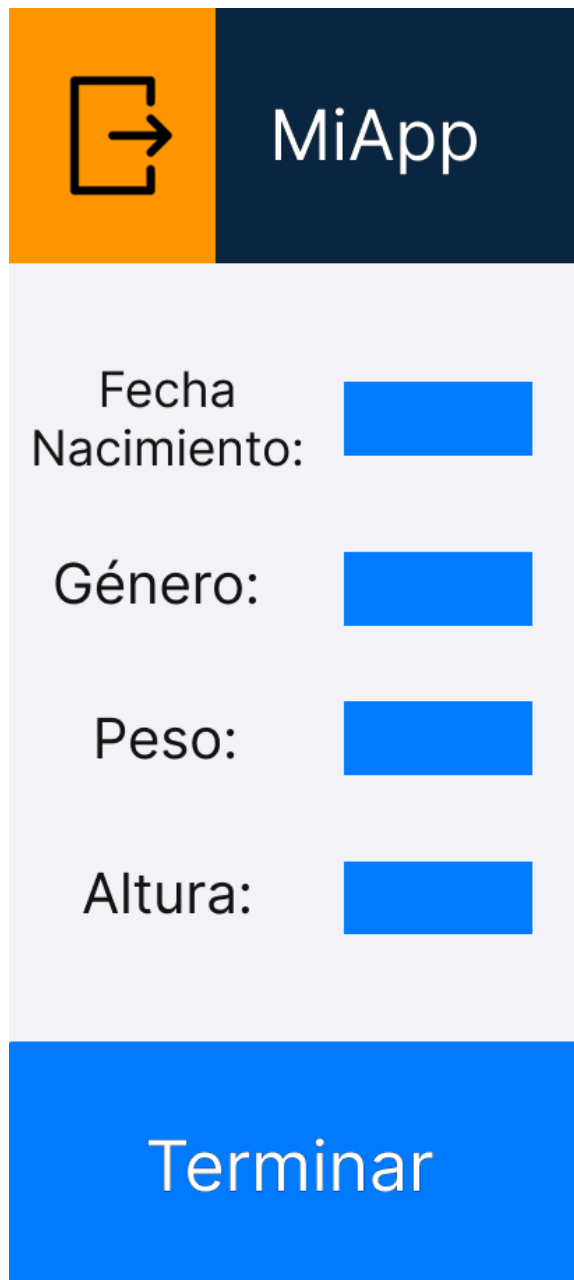
Usuario

Contraseña

Repite contraseña

Seguir

Figura 3.5: Crear cuenta 1

 MiApp

Fecha Nacimiento:

Género:

Peso:

Altura:

Terminar

Figura 3.6: Crear cuenta 2

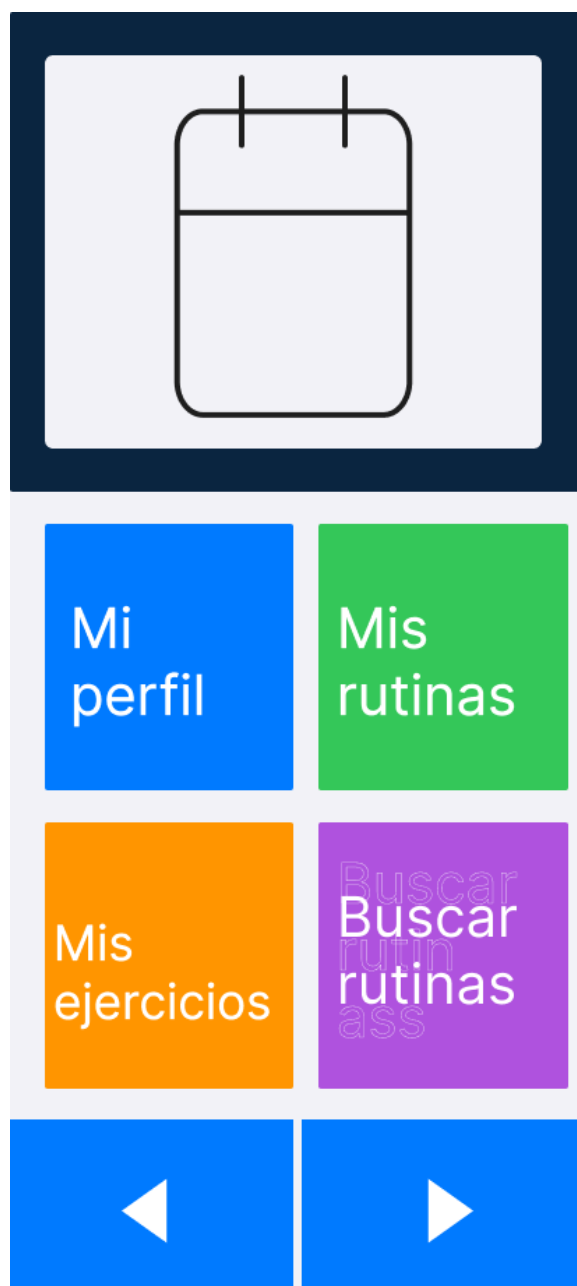


Figura 3.7: Menu principal

Lista ejercicios del usuario



Figura 3.8: Lista ejercicios

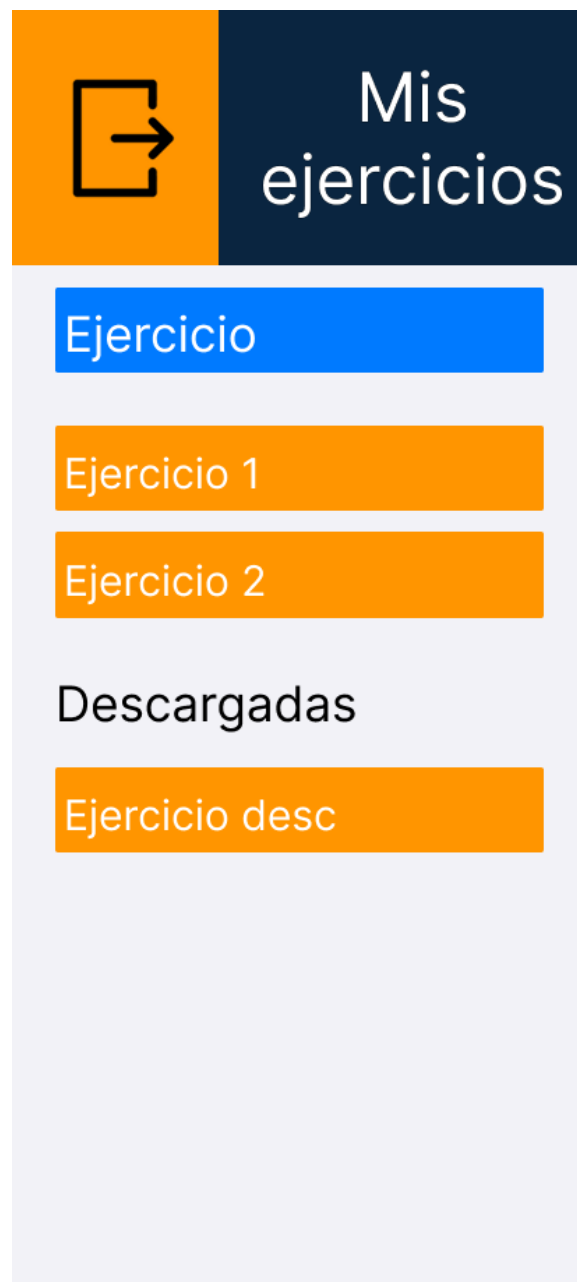


Figura 3.9: Lista ejercicios filtrada



	Nombre ejercicio	
<p>Marca actual: 7.4 repeticiones x 6kg</p> <p>Valoración actual: 4.5 / 5</p> <p>Graficar según tiempo</p> <p>Meta actual: 8 repeticiones x 6kg</p> <p>Sin cumplir</p> <p>Establecer meta</p>		

Figura 3.10: Datos ejercicio



Figura 3.11: Pop up graficar segun tiempo

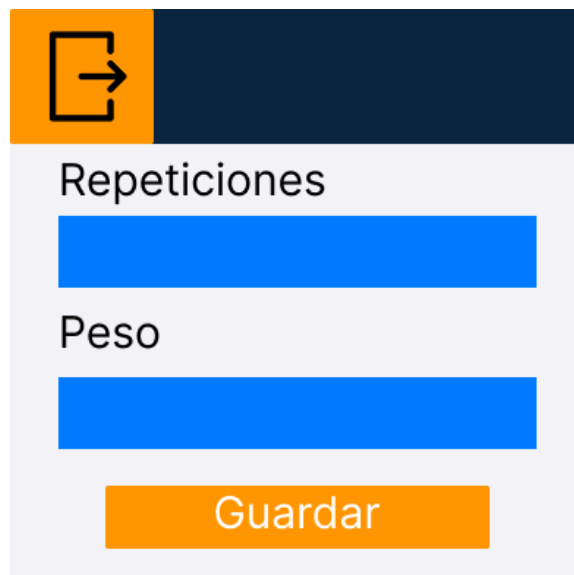
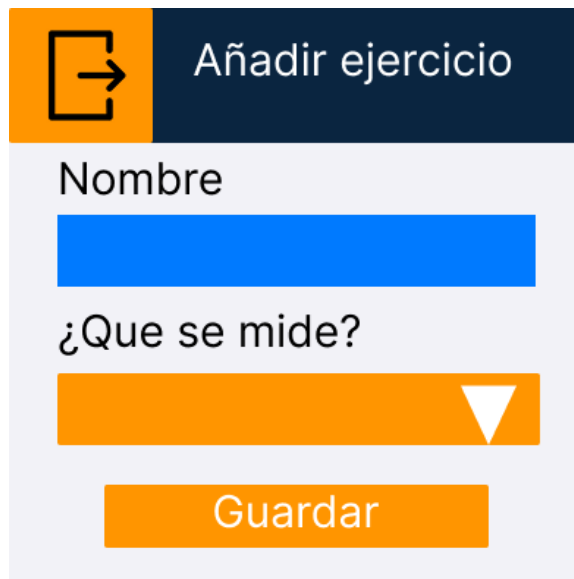
The image shows a square icon divided into two horizontal sections. The top section is orange and contains a white outline of a window with an arrow pointing to the right. The bottom section is dark blue and contains a white line graph with five data points connected by lines, showing an overall upward trend. Below the line graph are five dark blue vertical bars of varying heights, representing a bar chart.

Figura 3.12: Pop up establecer meta



Añadir ejercicio

Nombre

¿Que se mide?

Guardar

The image shows a mobile application pop-up for creating a new exercise. It has a dark blue header with a white icon of a square with an arrow pointing right, and the text 'Añadir ejercicio'. Below the header is a light gray form area. The first field is labeled 'Nombre' and has a blue input box. The second field is labeled '¿Que se mide?' and has an orange input box with a white downward-pointing triangle on the right side, indicating a dropdown menu. At the bottom of the form is an orange button with the text 'Guardar'.

Figura 3.13: Pop up crear ejercicio

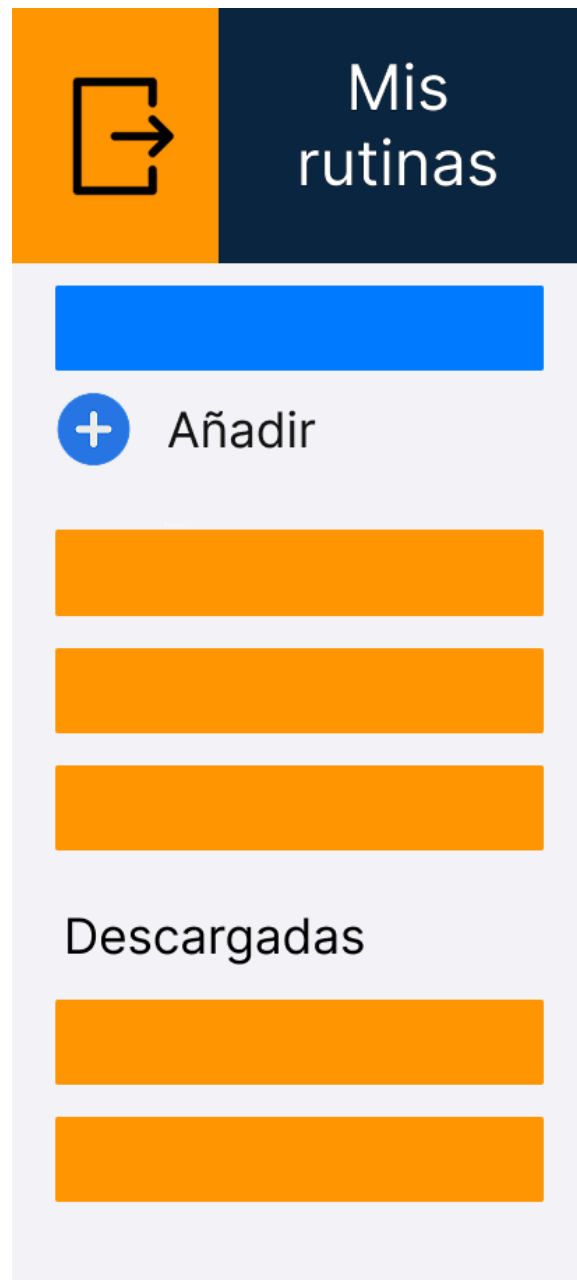


Figura 3.14: Lista rutinas

Pop up datos de un ejercicio en entrenamiento

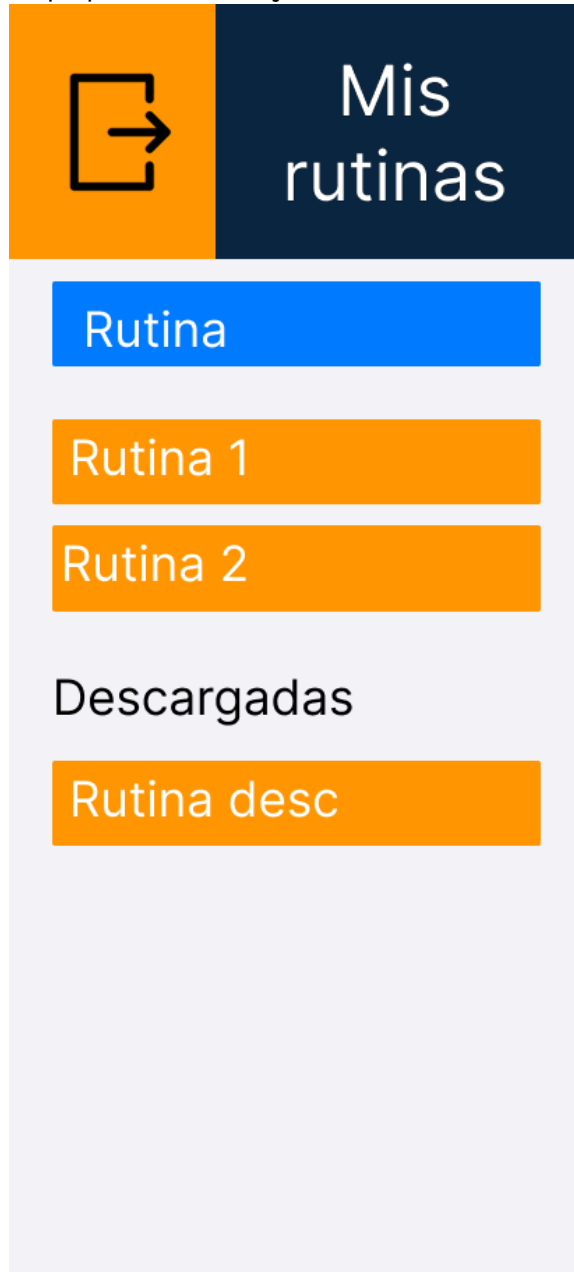


Figura 3.15: Lista rutinas filtradas



Figura 3.16: Datos rutina modificable

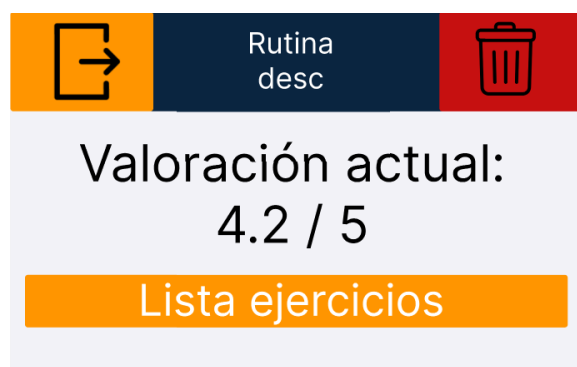


Figura 3.17: Datos rutina no modificable



Figura 3.18: Modificar ejercicios rutina

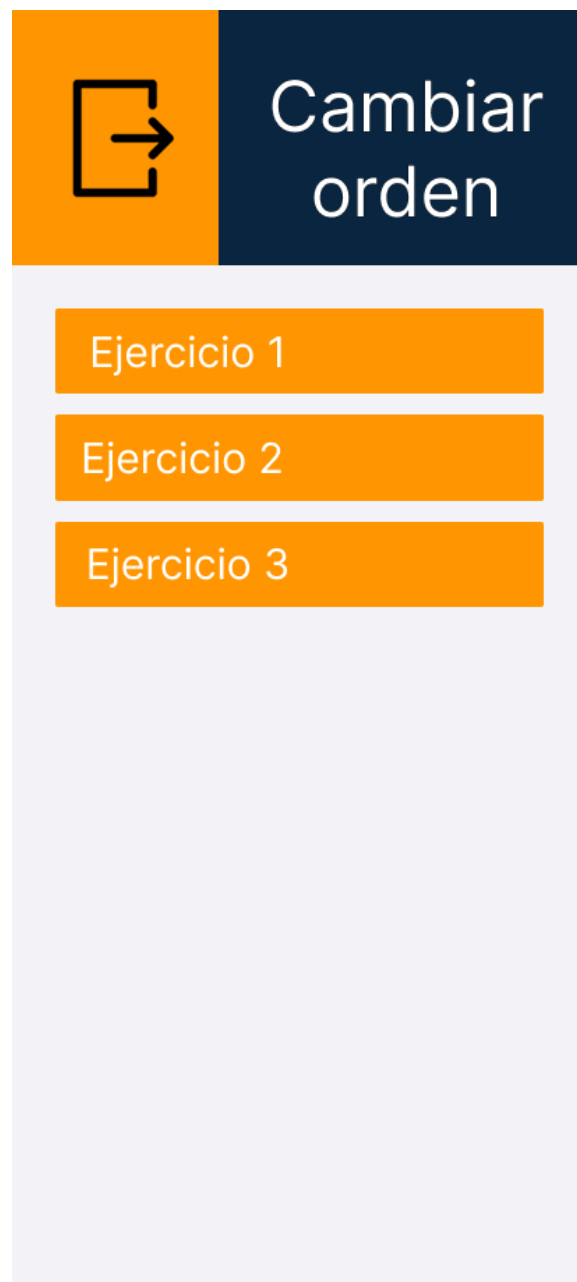


Figura 3.19: Cambiar orden ejercicios rutina

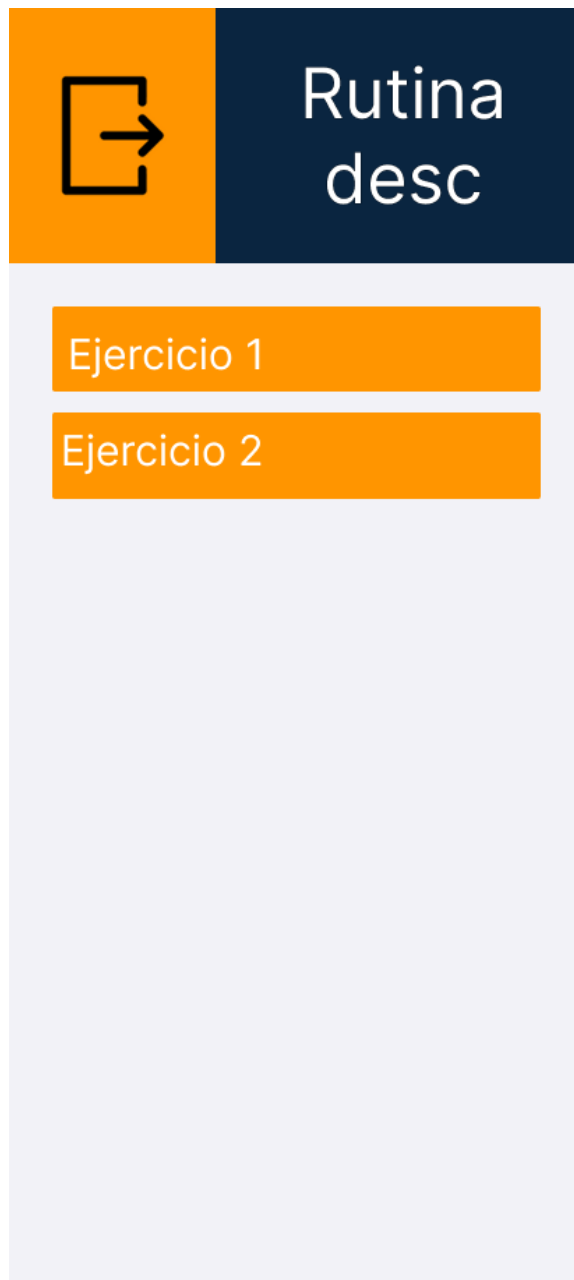
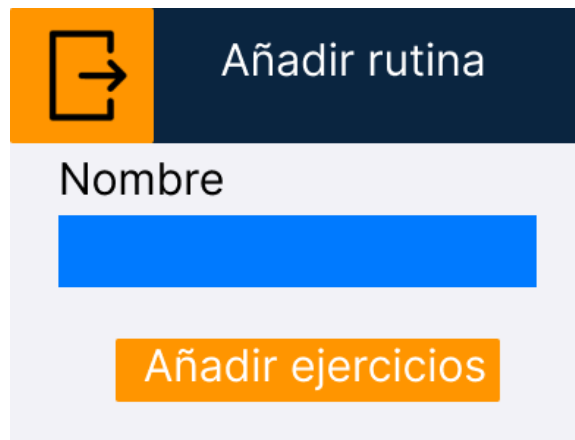


Figura 3.20: Ejercicios rutina descargada



A pop-up form for creating a routine. It features a dark blue header with a white icon of a square with an arrow pointing right, and the text "Añadir rutina". Below the header is a light gray section with the label "Nombre" and a blue rectangular input field. At the bottom of this section is an orange button with the text "Añadir ejercicios".

Figura 3.21: Pop up crear rutinas

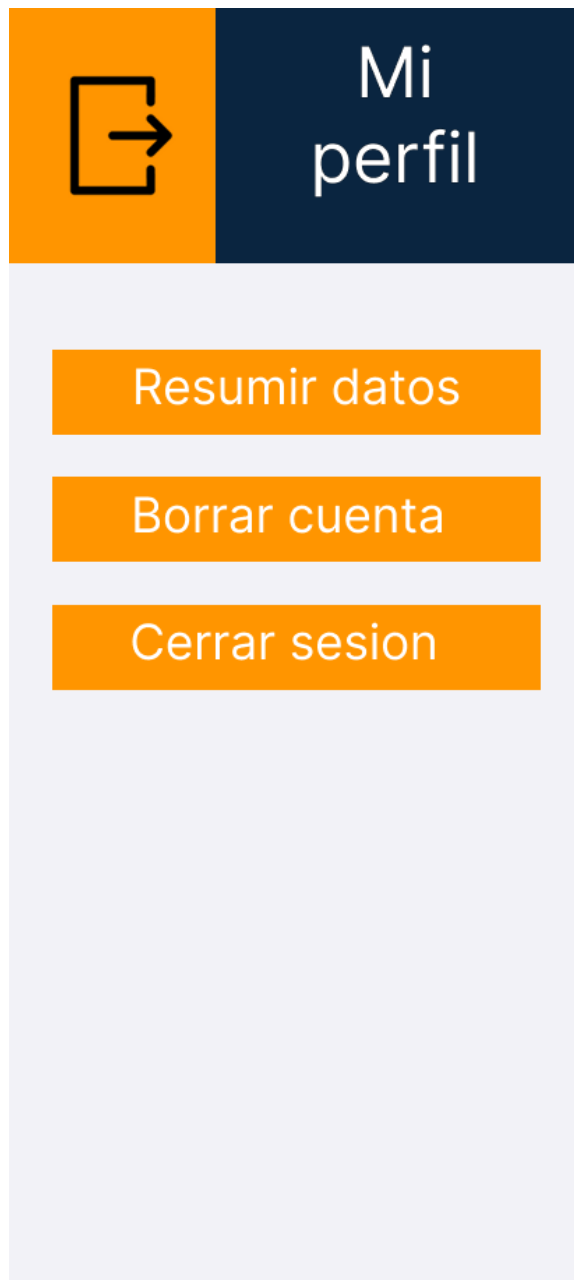


Figura 3.22: Opciones de perfil usuario



Figura 3.23: Pop up de confirmacion

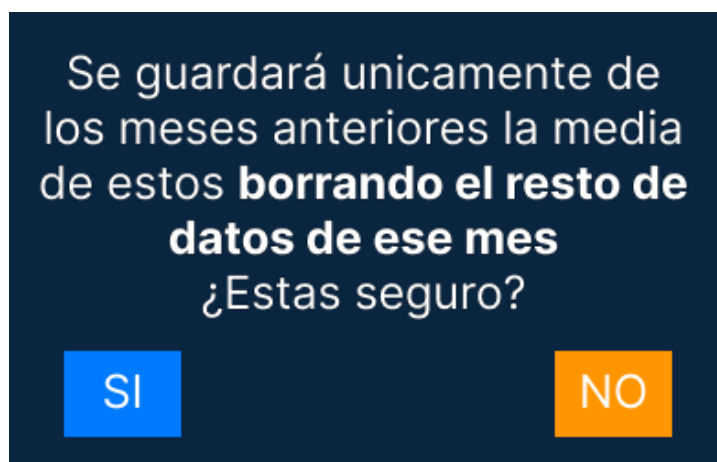


Figura 3.24: Pop up de confirmacion resumir datos

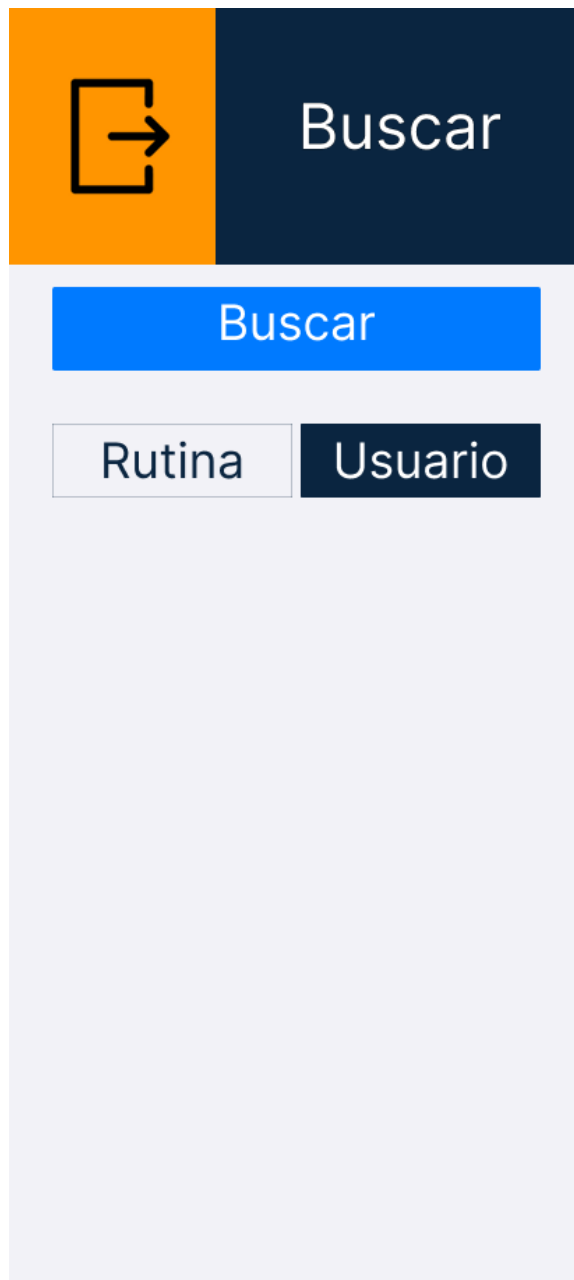


Figura 3.25: Buscar usuario



Figura 3.26: Buscar usuario filtrado

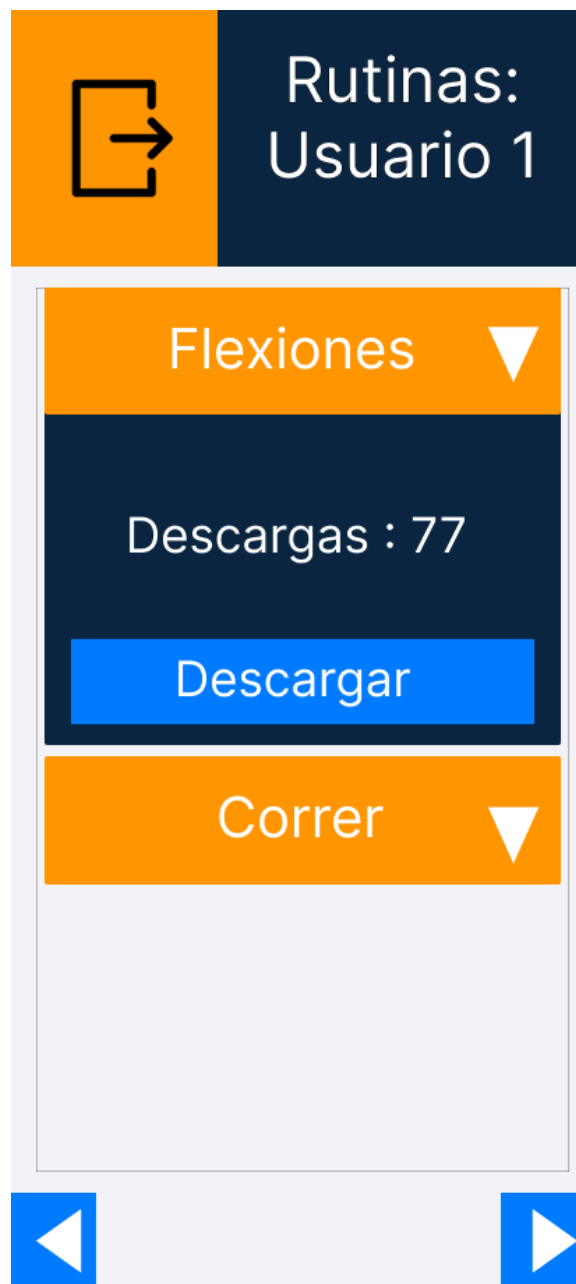


Figura 3.27: Rutinas de un usuario

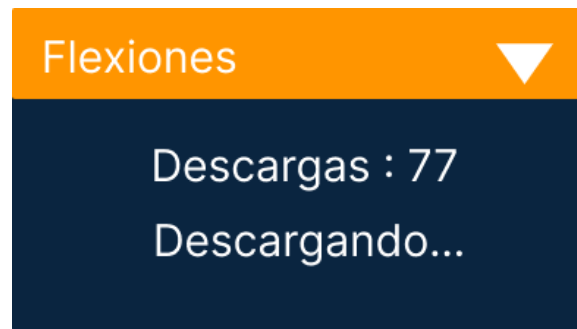


Figura 3.28: Widget descargar rutina

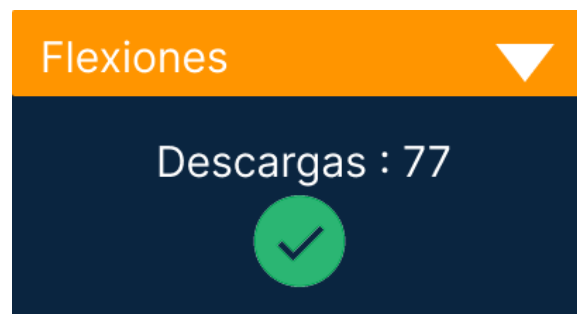


Figura 3.29: Widget rutina descargada



Figura 3.30: Buscar rutinas por su nombre filtrada

Entrenamiento (Una vez seleccionada una fecha en el calendario del menu principal saldría la siguiente ventana)

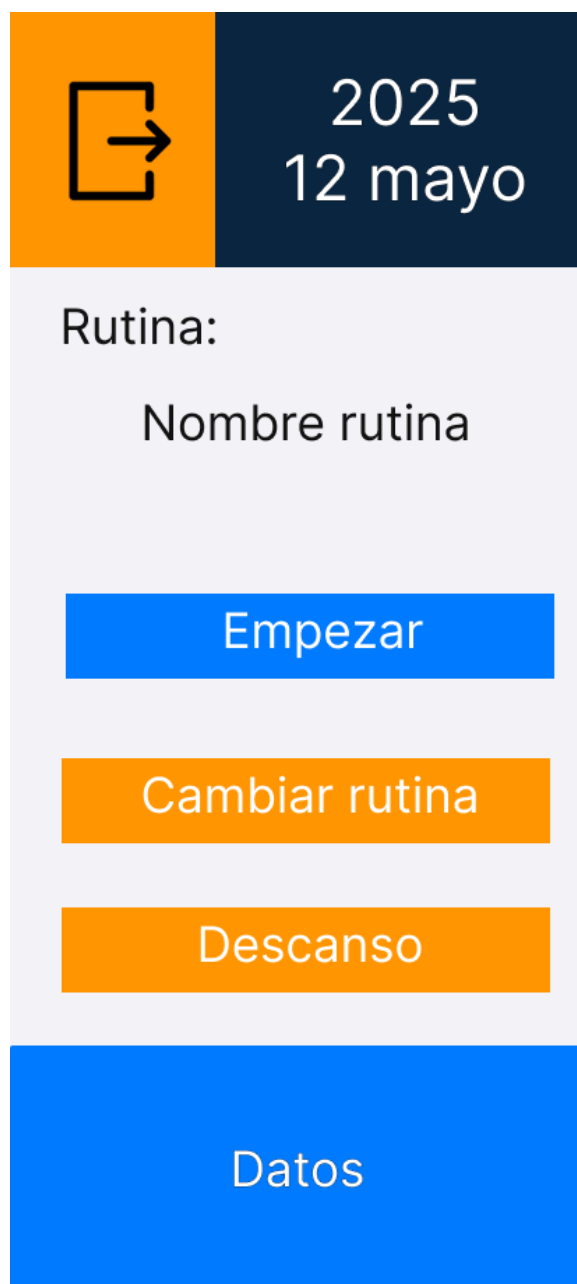


Figura 3.31: Entrenamiento de un día determinado

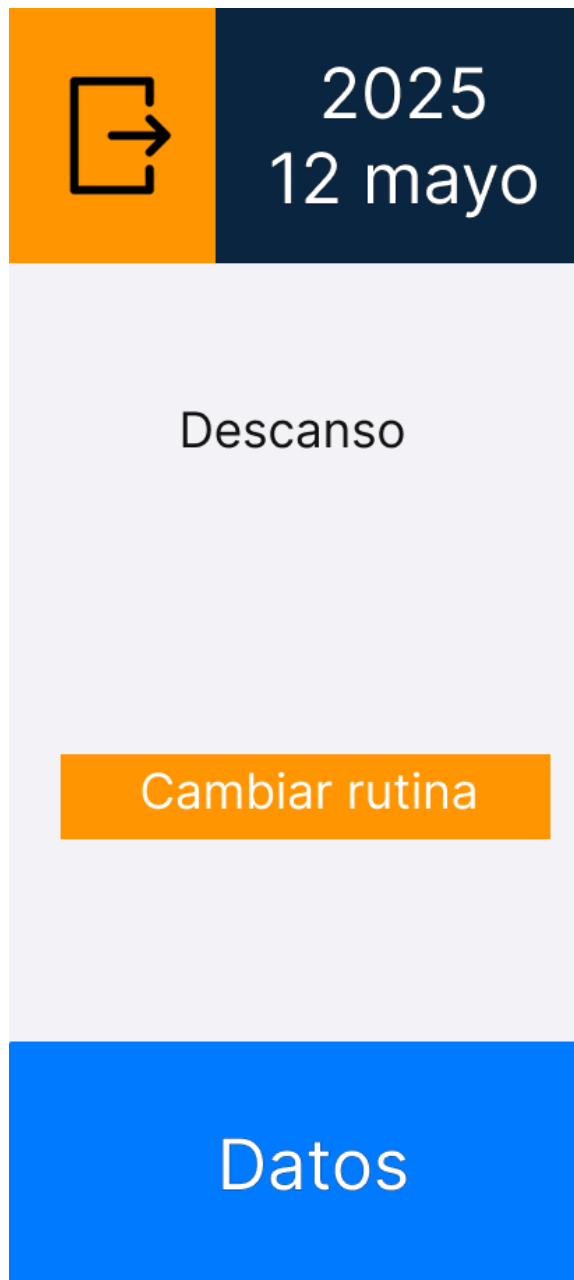


Figura 3.32: Descanso en un día determinado

Si seleccionamos datos de una fecha saldría esto

The image shows a mobile application interface with a dark blue header bar. On the left of the header is an orange square containing a white icon of a door with an arrow pointing right. To the right of the icon, the text '2025' and '12 mayo' is displayed in white. Below the header is a light gray area. In this area, the text 'Peso corp: sin ingresar' is shown. Below this text is an orange button with the word 'Ingresar' in white. Further down, the text 'Composicion corporal:' is displayed, followed by 'Músculo: X% Grasa: X% Osea: X%'. Below this text is another orange button with the word 'Ingresar' in white.

Figura 3.33: Frame 29.png

La siguiente pantalla sale al comenzar el entrenamiento

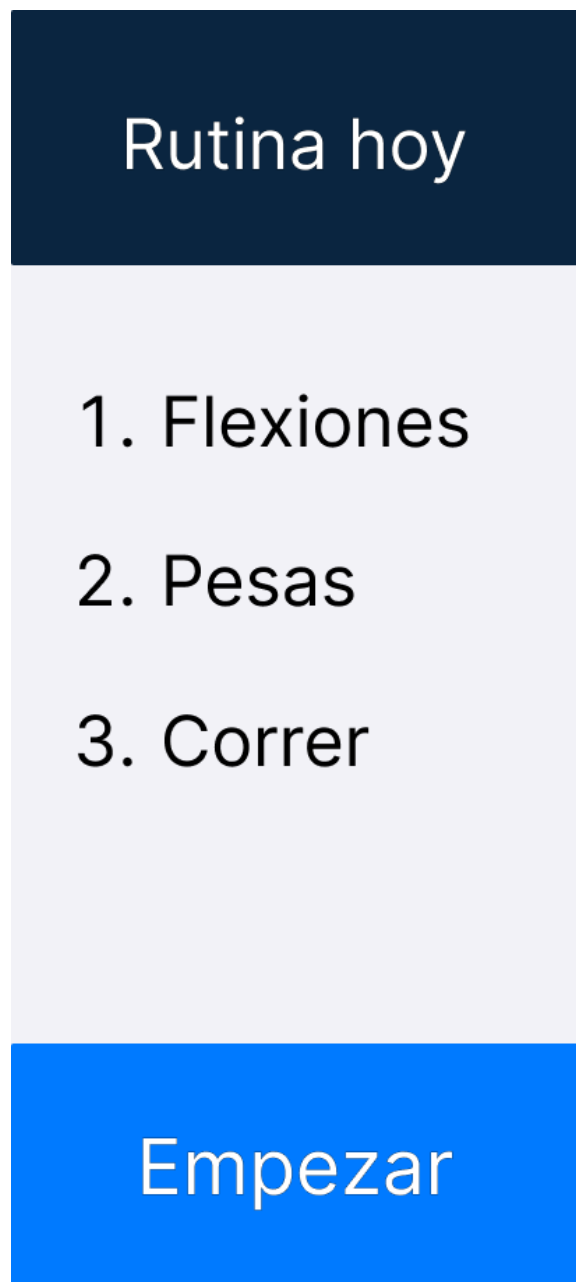


Figura 3.34: Lista ejercicios del entrenamiento actual

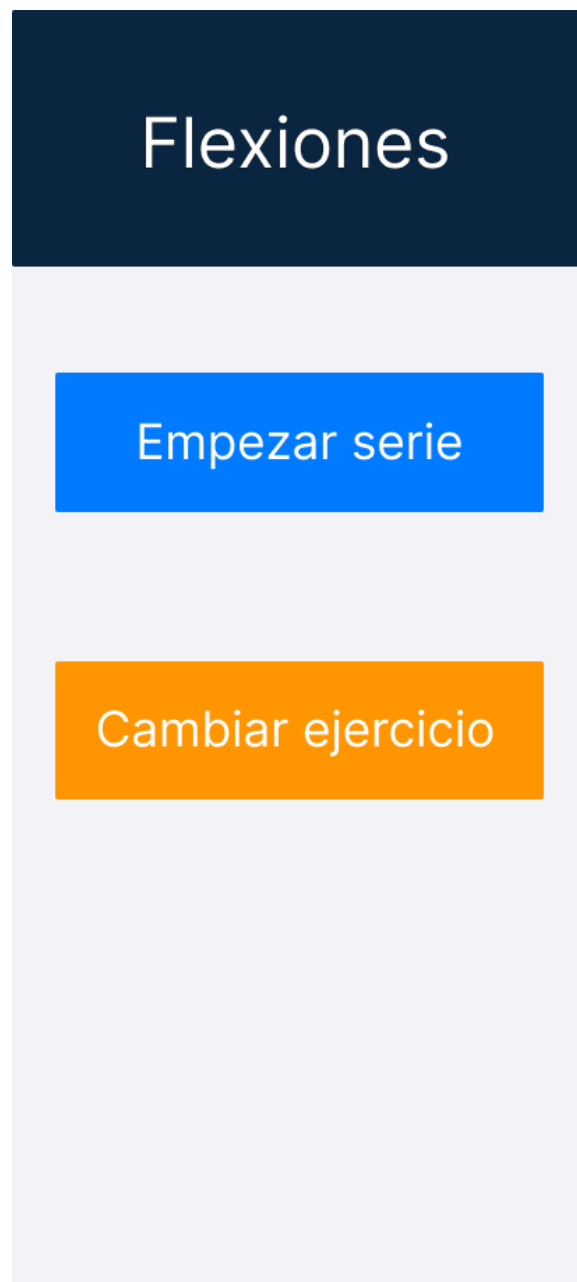


Figura 3.35: Primera serie flexiones

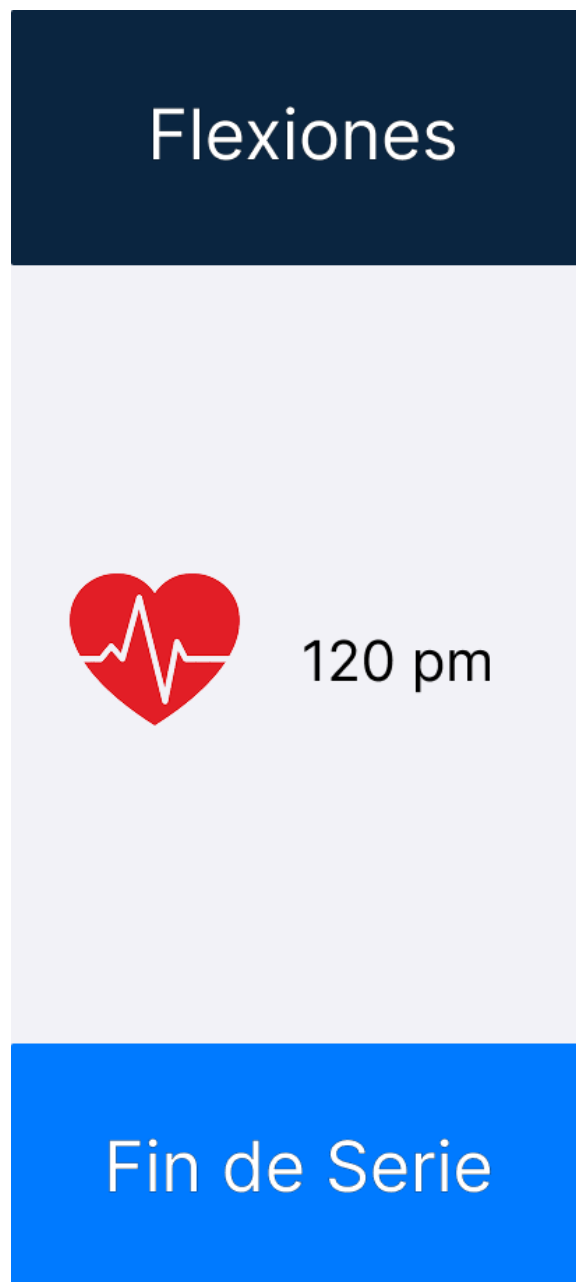


Figura 3.36: Realizando flexiones

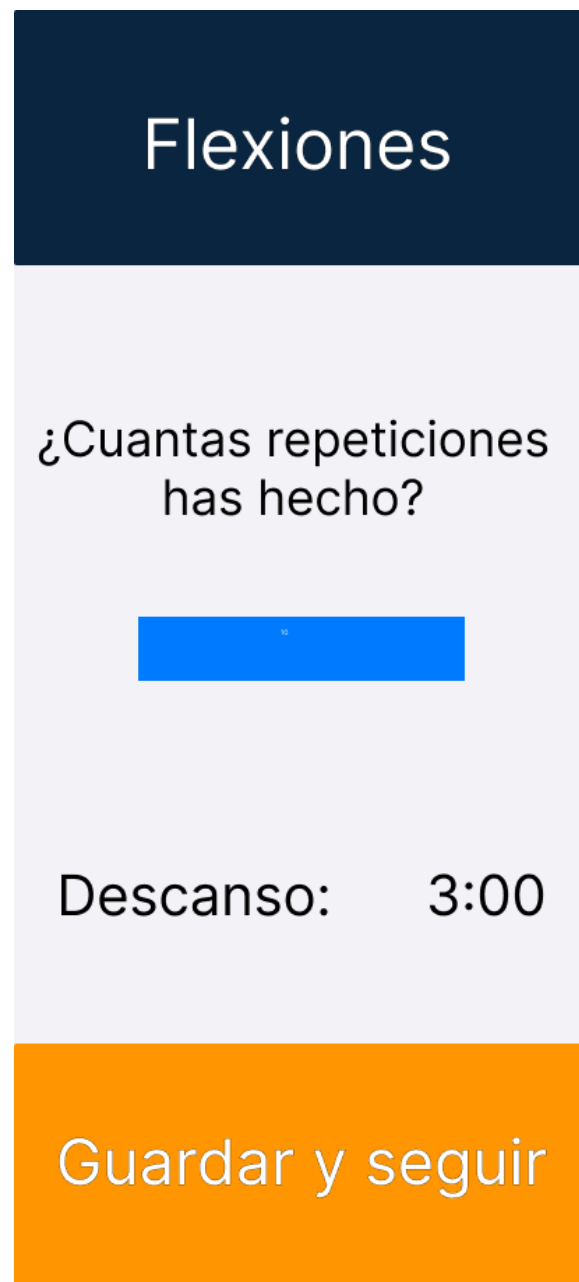


Figura 3.37: Fin de la serie(Descanso no completado)

Flexiones

¿Cuántas repeticiones
has hecho?

Descanso: 0:00

Guardar y seguir

Figura 3.38: Fin de la serie(Descanso completado)

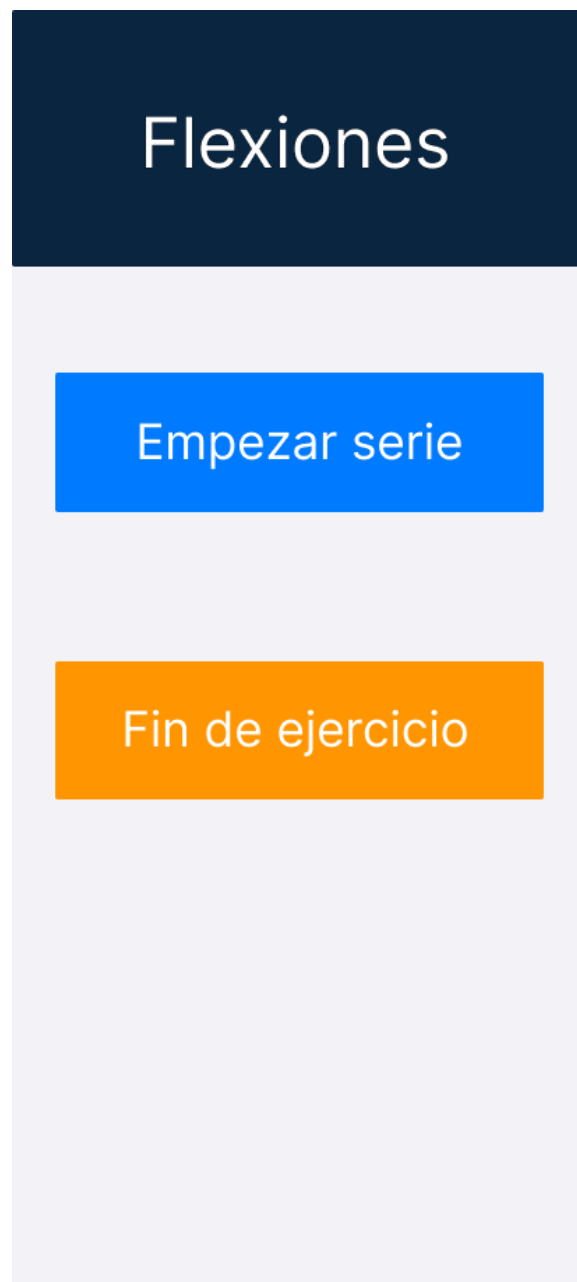


Figura 3.39: Acabar ejercicio o añadir serie

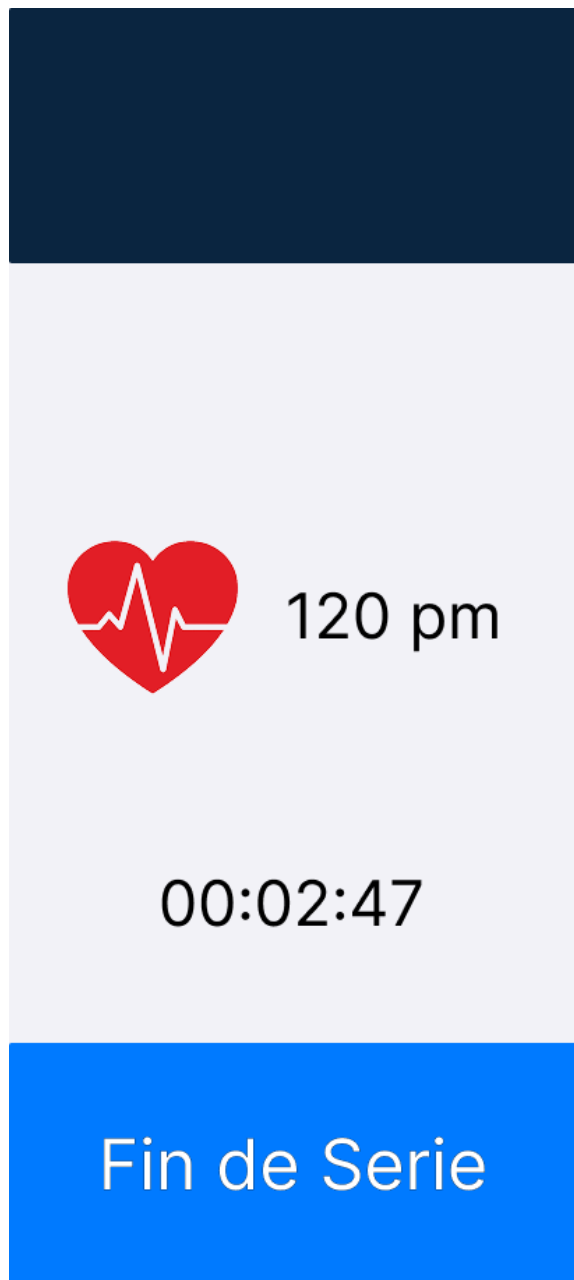


Figura 3.40: Realizando ejercicio midiendo tiempo

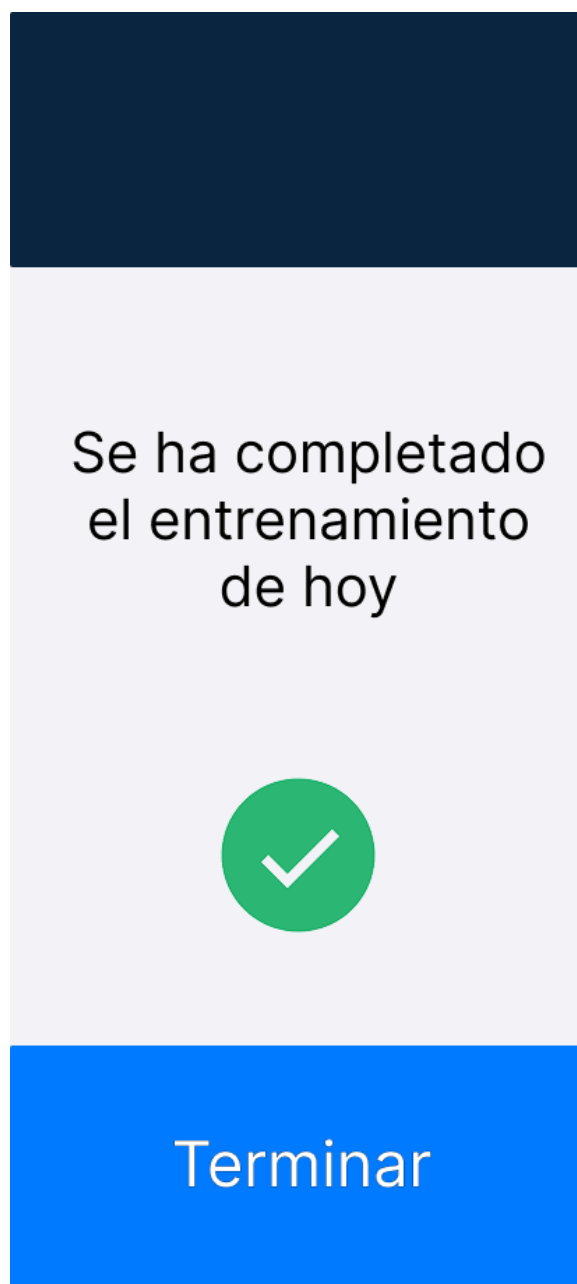


Figura 3.41: Fin entrenamiento



Figura 3.42: Metas cumplidas

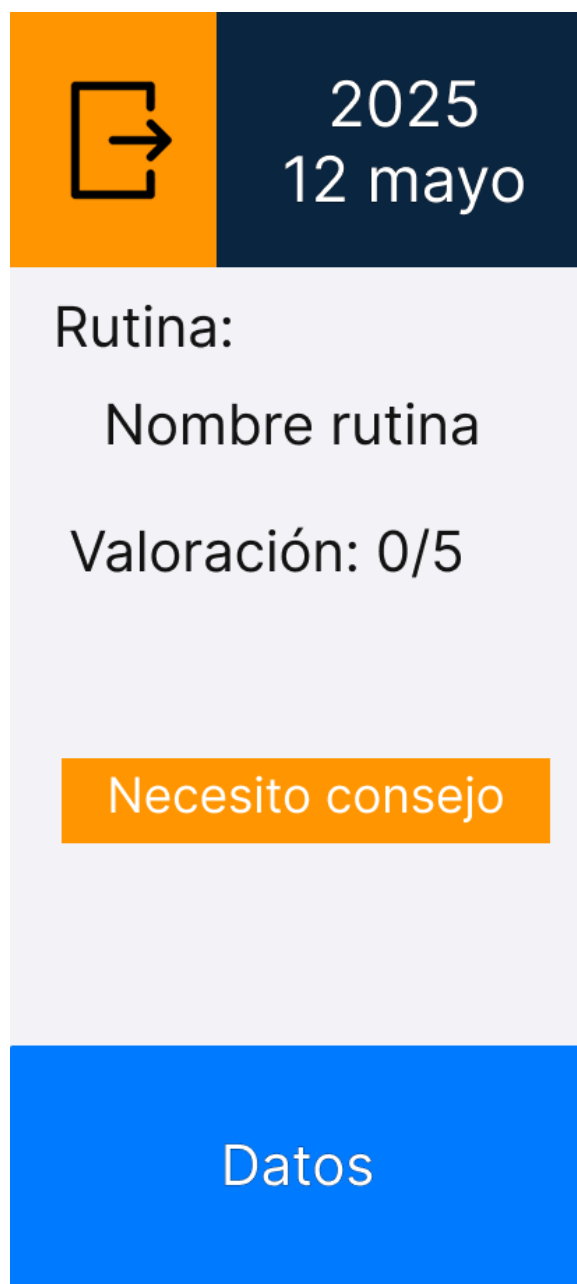


Figura 3.43: Datos entrenamiento terminado

El chat de la IA se abre cuando se le da a la opción necesito consejo de un entrenamiento realizado

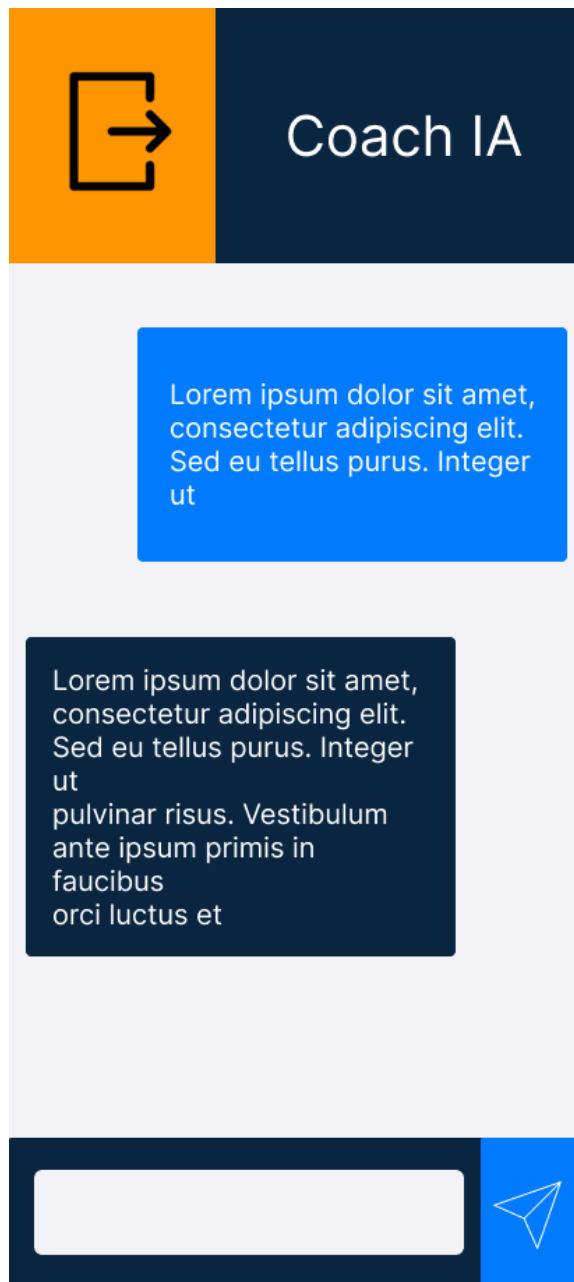


Figura 3.44: Chat con IA

Si le das a datos de un entrenamiento terminado

The image shows a mobile application interface for recording training data. At the top, there is a header bar with an orange square on the left containing a white icon of a document with an arrow pointing right, and a dark blue square on the right containing the text "2025" and "12 mayo" in white. Below the header, there is a light gray background area containing two orange rectangular buttons labeled "Ejercicio 1" and "Ejercicio 2" in white text. Below these buttons is a horizontal bar with a light gray center and blue squares on either side containing white left and right arrow icons. At the bottom, there is a dark blue section containing the text "Peso corp: sin ingresar" in white. Below this text is an orange button labeled "Ingresar" in white. Further down, the text "Composicion corporal:" is displayed in white, followed by "Músculo: X% Grasa: X% Osea: X%". At the very bottom of this section is another orange button labeled "Ingresar" in white.

Figura 3.45: Datos detallados entrenamiento terminado

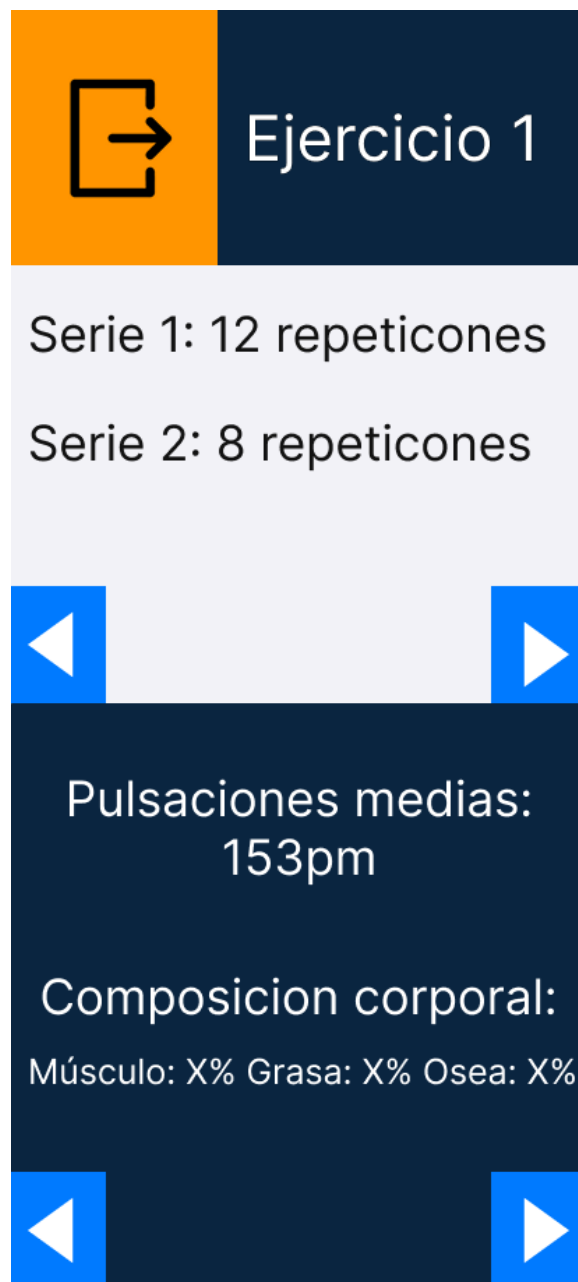


Figura 3.46: Pop up datos de un ejercicio en entrenamiento

Estos diseños pueden cambiar y sufrir alteraciones debido a los sprints reviews y correcciones de accesibilidad. De todas formas, cada cambio será informado con su motivo.

3.9.1.1. Sprint Review 0

Durante esta revisión de sprint se realizaron mejoras de diseño, como la incorporación de iconos en todos los botones para mejorar la accesibilidad. Se revisaron también los títulos de las ventanas, cambiando aquellos que no eran lo suficientemente claros.

Además, se añadieron nuevas historias al *product backlog*, incorporando o desglosando funcionalidades:

- SCRUM-29: Copiar rutina
- SCRUM-30: Añadir meta por parámetro
- SCRUM-31: Solicitar permiso al usuario antes de enviar datos a la IA
- SCRUM-32: Enviar datos del entrenamiento actual y anteriores a la IA
- SCRUM-33: Conectar/Desconectar con la IA

El backlog quedaría de la siguiente manera:

ID	Descripción	Prioridad (MoSCoW)
SCRUM-1	Registrar peso por día	C
SCRUM-2	Establecer peso objetivo	W
SCRUM-3	Insertar/Borrar/Modificar ejercicio de la lista de ejercicios	M
SCRUM-4	Buscar rutina en la lista del usuario	M
SCRUM-5	Insertar/Borrar/Modificar rutina	M
SCRUM-6	Hacer gráfica en base a las marcas obtenidas	C
SCRUM-7	Revisar datos para ver si el descanso es necesario	W
SCRUM-8	Enseñar datos de una rutina a descargar	S
SCRUM-9	Compartir mi rutina	M
SCRUM-10	Valorar el entrenamiento en base a la marca actual y la meta del usuario	W
SCRUM-11	Monitorizar pulso en tiempo real	C
SCRUM-12	Medir pulso en reposo y compararlo con datos de ejercicios	W
SCRUM-13	Avisar de anomalías en el pulso de forma suave	C
SCRUM-14	Obtener calorías quemadas	W
SCRUM-15	Comprobar el equilibrio nervioso del usuario	W
SCRUM-16	Realizar el flujo del entrenamiento	M
SCRUM-17	Conectar con la IA para iniciar diálogo	C

Figura 3.47: Backlog prioridades MoSCoW

SCRUM-18	Crear/Borrar usuario	M
SCRUM-19	Resumir datos	W
SCRUM-20	Iniciar/Cerrar sesión	M
SCRUM-21	Medir SpO2	W
SCRUM-22	Interpretar constantes	C
SCRUM-23	Buscar ejercicios en la lista de ejercicios	M
SCRUM-24	Pop up de confirmación	M
SCRUM-25	Implementar menú principal	M
SCRUM-26	Detectar metas cumplidas en los ejercicios despues del entrenamiento	S
SCRUM-27	Buscar rutinas para descargar	M
SCRUM-28	Implementar calendario	M
SCRUM-29	Copiar rutina	C
SCRUM-30	Añadir meta por parámetro	C
SCRUM-31	Solicitar permiso al usuario antes de enviar datos a la IA	M
SCRUM-32	Enviar datos del entrenamiento actual y anteriores a la IA	S
SCRUM-33	Conectar/Desconectar con la IA	M

Figura 3.48: Backlog prioridades MoSCoW

3.9.2. Iteración 1

La lista de historias de la primera iteracion queda de la siguiente manera:

- SCRUM-23: Buscar ejercicio en lista de ejercicios
- SCRUM-3: Insertar/Borrar/Modificar ejercicio de la lista de ejercicios
- SCRUM-24: Pop up de confirmación
- SCRUM-30: Añadir meta por parámetro

A continuación especificaremos las tareas y pruebas de cada historia, y los resultados asociados a cada una.

SCRUM-23: Buscar ejercicio en lista de ejercicios
Tareas:
1. Implementar la BD local
2. Implementar algoritmo de búsqueda por nombre
3. Implementar boceto de IU fig. 3.8
Pruebas de aceptación:
■ si no existe ejercicio buscado, no sale nada en pantalla
■ si existe ejercicio buscado, sale un acceso en pantalla

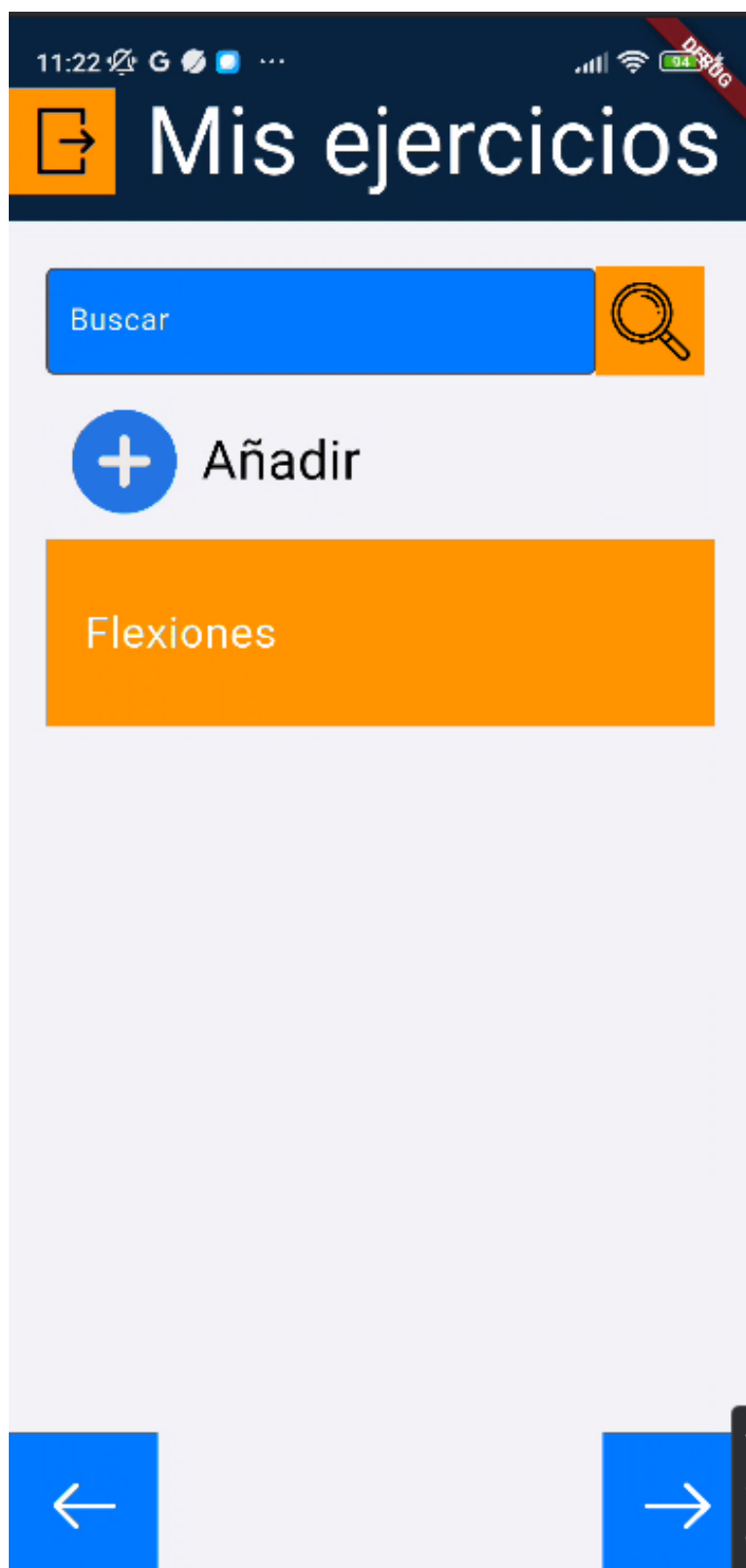


Figura 3.49: Lista ejercicios

Una de las dificultades de esta funcionalidad fue que esta pantalla debería funcionar como una plantilla, es decir, se han de pasar las funciones de consultas y la clase construiría una lista pasase lo que se pasase, de esta forma se reciclaría mucho código ahorrando tiempo. Lo que se hizo fue una ingeniería inversa, se desarrollo la pantalla forma específica para los ejercicios y cuando funcionaba correctamente, se fue abstrayendo el código hasta obtener la clase ListaBusquedaAniadir.

Otro problema encontrado durante las pruebas fue la actualización de la lista, se tuvieron muchos problemas dado que en flutter si se tiene una instancia de un widget llamemoslo A, que forma parte de otro llamado B, si yo llamo a una actualización de la IU desde B porque ha cambiado el contenido se genera una excepción. Como solución, se optó por solo usar la función setState (Usada para actualizar el contenido) dentro del código de la plantilla y cada vez que fuera necesario hacer una actualización desde B recargar otra vez la pantalla entera.

Las pruebas realizadas fueron de caja negra, ir añadiendo elementos para que se hagan visibles en la IU e ir comprobando navegabilidad del menu, el algoritmo de búsqueda y que se actualizaba correctamente al añadir algún objeto. Pasó todas las pruebas.

SCRUM-3: Insertar/Borrar/Modificar ejercicio de la lista de ejercicios
<p>Tareas:</p> <ol style="list-style-type: none"> 1. Implementar la BD local 2. Implementar las funciones para las consultas 3. Implementar las funciones para las modificaciones 4. Implementar las funciones para el borrado 5. Implementar boceto de la IU fig. 3.10, para borrar/modificar desde ahí <p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> ■ si se borra un ejercicio, no sale en la lista ■ si se modifica, sale con los datos modificados ■ si se inserta, sale en la lista el ejercicio ■ se enseñan los datos del ejercicio de forma correcta

The screenshot shows a mobile application interface for creating an exercise. At the top, there is a dark blue header with a white icon of a square with an arrow pointing right, followed by the text "Crear ejercicio". The status bar at the very top shows the time 11:22, signal strength, Wi-Fi, and battery level at 94%. Below the header, the form is divided into sections. The first section is titled "Nombre" and contains a blue input field with the placeholder text "Nombre". The second section is titled "Parámetros" and contains four orange buttons labeled "Repeticiones", "Peso", "Tiempo", and "Distancia". Below these buttons is a large blue text area with the placeholder text "Descripcion". At the bottom of the form is a large orange button labeled "Guardar".

11:22 G 94%

Crear ejercicio

Nombre

Nombre

Parámetros

Repeticiones Peso Tiempo Distancia

Descripcion

Guardar

Figura 3.50: Crear ejercicios

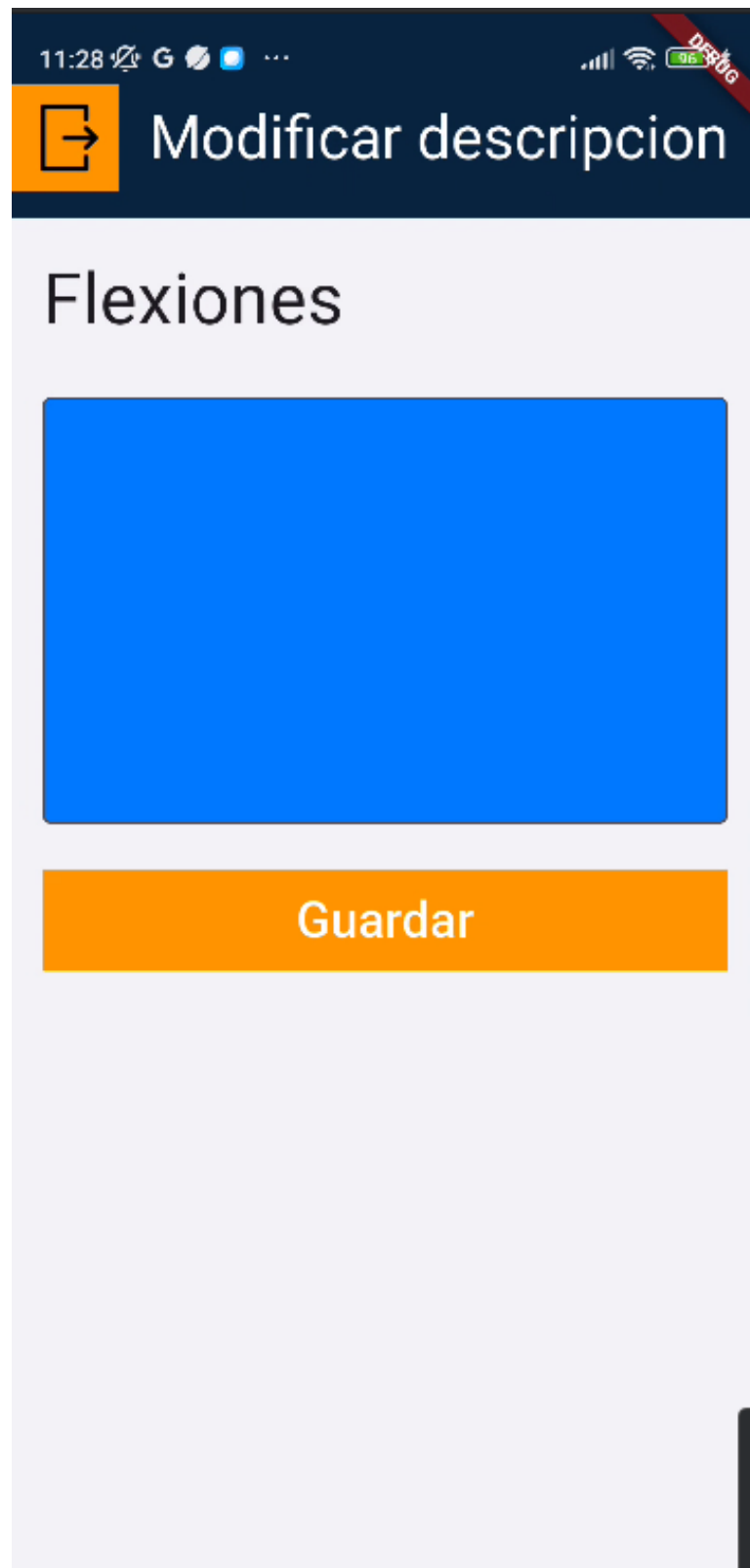
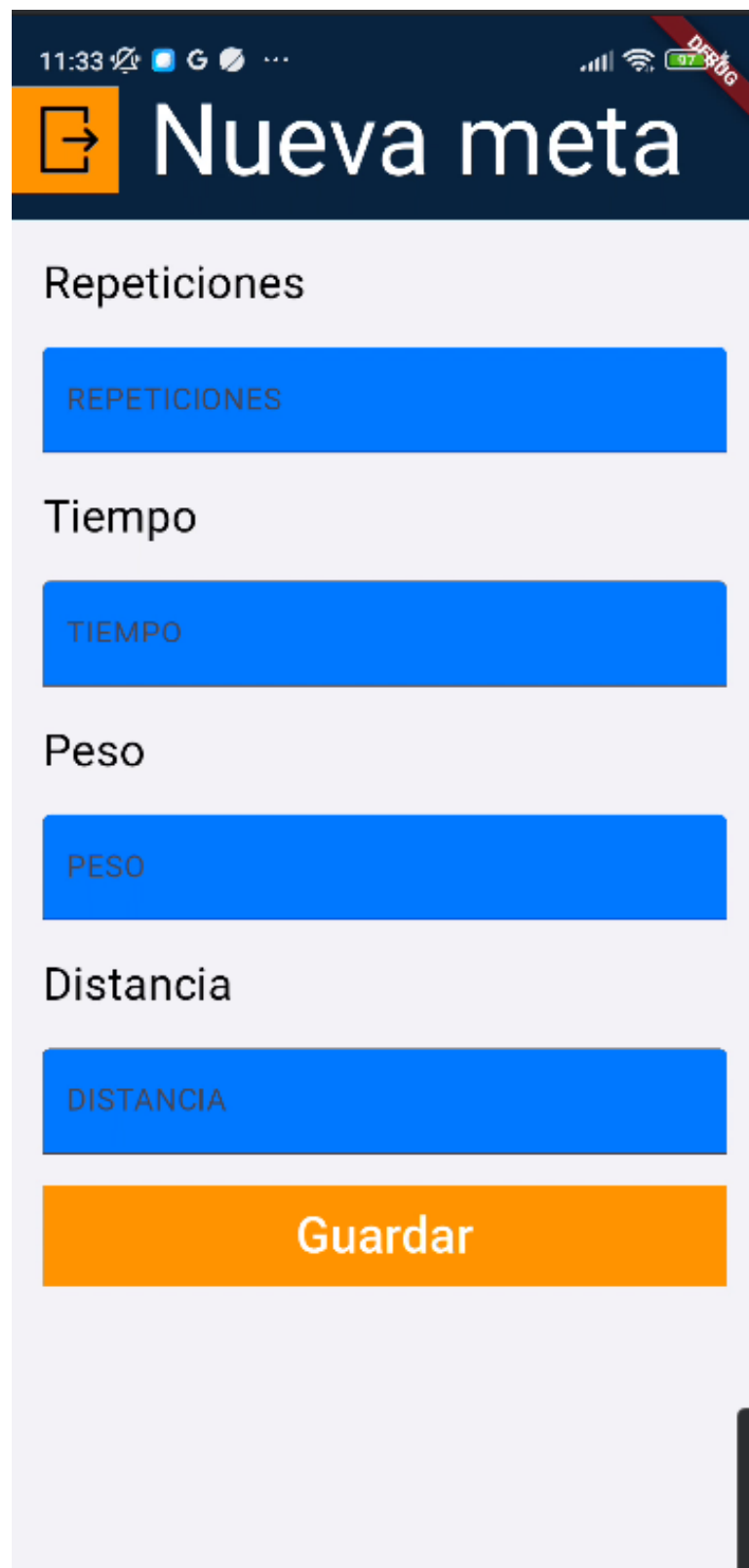


Figura 3.51: Modificar descripcion ejercicios

Las pruebas realizadas para la SCRUM-3, fueron de caja negra, ir haciendo inserciones y borrados en la BD local para después hacer consultas y comprobar si los resultados eran correctos. También se comprobó su correcto funcionamiento con el SCRUM-3, corroborando la correcta actualización de los contenidos. Pasó todas las pruebas.

SCRUM-30: Añadir meta por parámetro
Tareas:
1. Implementar la BD local
2. Implementar las funciones para modificar metas
Pruebas de aceptación:
■ si existe una meta determinada para ese ejercicio y se quiere insertar una, sustituir la actual



The image shows a mobile application interface for setting a new goal. At the top, there is a dark blue header with a white icon of a square with an arrow pointing right, followed by the text "Nueva meta". Below the header, there are four input fields, each with a label above it: "Repeticiones", "Tiempo", "Peso", and "Distancia". Each input field is a blue rectangle with the corresponding label in all caps inside it. At the bottom of the form is a large orange button with the text "Guardar" in white. The background of the form is light gray. The top of the screen shows a status bar with the time 11:33, signal strength, Wi-Fi, and battery level (97%).

Figura 3.52: Nueva meta

El problema de esta iteración fue el como distinguir internamente, si un usuario quiere superar con un valor mayor o menor su meta, es decir, si un usuario quiere aumentar el peso con el que hace un ejercicio o disminuirlo. Al final optó por que solo se pueda aumentar.

Las pruebas realizadas se basaron en añadir valores nuevos de metas tanto erroneos como válidos, para ver como se comportaban las soluciones implementadas a las erratas que pueda introducir el usuario. Pasó todas las pruebas.

SCRUM-24: Pop up de confirmación
Tareas:
1. Implementar la confirmación para que pueda ser usado en pantallas distintas
2. Implementar boceto de IU fig. 3.23
Pruebas de aceptación:
■ si se ha seleccionado una opcion, se devuelve la opcion seleccionada



Figura 3.53: PopUp Confirmacion

Es un pop up que se usará siempre que se requiera una confirmación ,se implementó para que pueda ser usado en cualquier contexto de la app, pasó las pruebas.

3.9.2.1. Estado de la BD local

Cabe aclarar que la BD local es la memoria persistente de la app que solo existe en el dispositivo que esta instalada la app, se le llama BD local porque tiene una estrucura tipo SQL.

Al finalizar esta iteración, el estado de la BD queda tal y como se muestra en la figura fig. 3.54

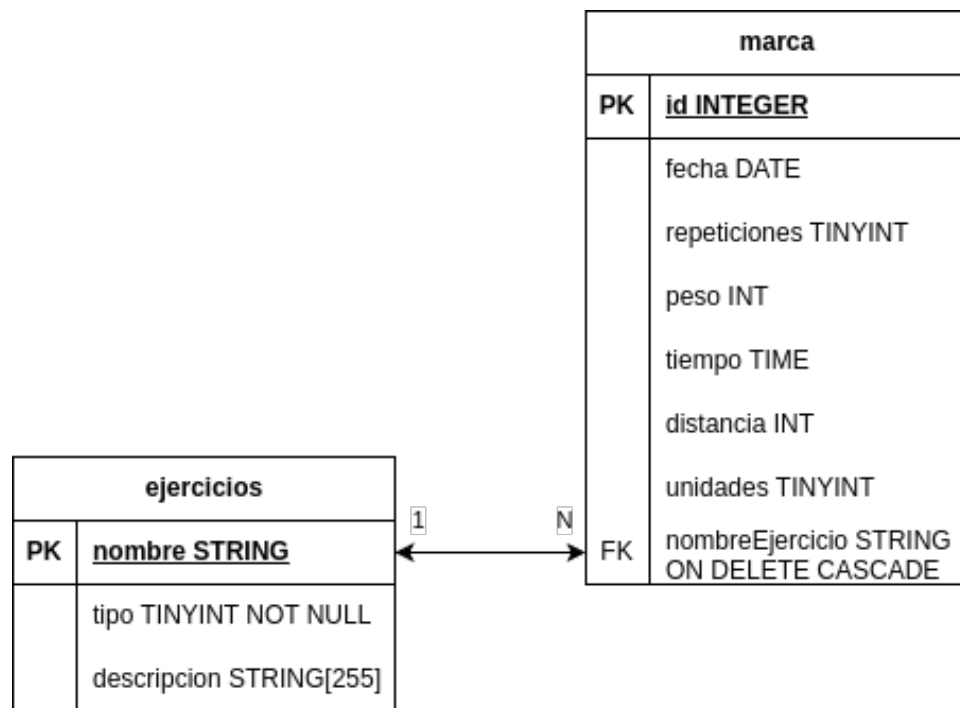


Figura 3.54: BD local iteracion 1

3.9.2.2. Sprint Review 1

Las primeras iteraciones tienden a ser más lentas debido a la fase inicial del desarrollo. No se completó la totalidad del sprint; quedó pendiente la subtask de modificar ejercicio en la base de datos (SCRUM-3: Insertar/Borrar/Modificar ejercicio de la lista de ejercicios). Aun así, las expectativas son positivas, ya que se espera un aumento en la velocidad de desarrollo.

La pantalla en la que se enseñan los datos de un ejercicio se modificó para facilitar su lectura y entendimiento, dado que su anterior diseño era confuso.

Otra parte a modificar es la del pop up de confirmación, ya que debido a la poca accesibilidad de los pop ups se sustituirá por una pantalla independiente.

También se cambiaron todos los pop ups y se sustituyeron por pantallas completas:

- Los pop ups de confirmacion
- Crear ejercicios
- Modificar ejercicios
- Nueva meta en un ejercicio



Figura 3.55: Pantalla nueva

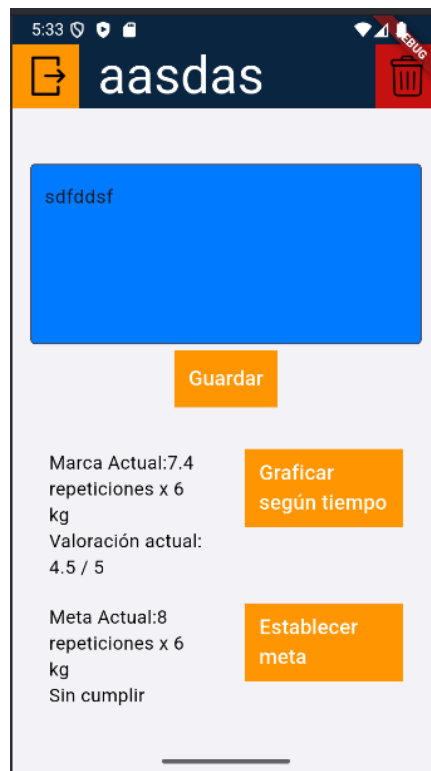


Figura 3.56: Pantalla antigua

Figura 3.57: Comparación entre la pantalla nueva y la anterior

- Crear rutinas
- Modificar rutinas
- Modificar ejercicios de una rutina
- Información de una rutina a descargar
- Modificar informacion del perfil del usuario

Por consecuencia las futuras funcionalidades en la que se mencionan los pop ups, pasaran a ser pantallas completas.

3.9.3. Iteración 2

Esta iteración se centró en el desarrollo del sistema de sesiones, la API de la aplicación, el backend básico del servidor y la sección de rutinas. Las historias

que incluye son las siguientes:

- SCRUM-18 Crear/Borrar mi usuario
- SCRUM-20 Iniciar/Cerrar sesión
- SCRUM-5 Insertar/Borrar/Modificar rutina
- SCRUM-4 Buscar rutina en la lista del usuario
- SCRUM-25 Implementar menú principal

También se terminará la historia no finalizada de la iteración anterior(SCRUM-3: Insertar/Borrar/Modificar ejercicio de la lista de ejercicios).

SCRUM-25 Implementar menú principal
Tareas:
1. Implementar el boceto IU fig. 3.7
2. <u>Añadir accesos a las funcionalidades actuales</u>
Pruebas de aceptación:
■ si la funcionalidad seleccionada no está implementada, no hacer nada
■ si la funcionalidad seleccionada está implementada, desplegar ventana de dicha funcionalidad



Figura 3.58: Menu Principal semifuncional

Las funcionalidades que no esten disponibles al ser pulsadas no hacen nada.

SCRUM-18 Crear/Borrar mi usuario
Tareas: 1. Implementar bocetos de IU fig. 3.5 ,fig. 3.6, fig. 3.3 y la parte necesaria de fig. 3.22 2. Implementar funciones de inserccion y borrado en el backend
Pruebas de aceptación: <ul style="list-style-type: none">■ si el usuario que se quiere crear ya existe, informar y pedir que se rellenen los credenciales otra vez■ si el usuario que se quiere crear no existe, crear usuario y pasar a la siguiente pantalla■ si cualquier dato requerido está en un formato incorrecto ,informar al usuario para que vuelva a rellenar estos■ si todos los datos están en el formato correcto ,seguir con la creación del usuario■ si se ha creado exitosamente el usuario, volver a la pantalla inicial■ si se borra el usuario, borrar el token en el dispositivo y sus credenciales en el backend

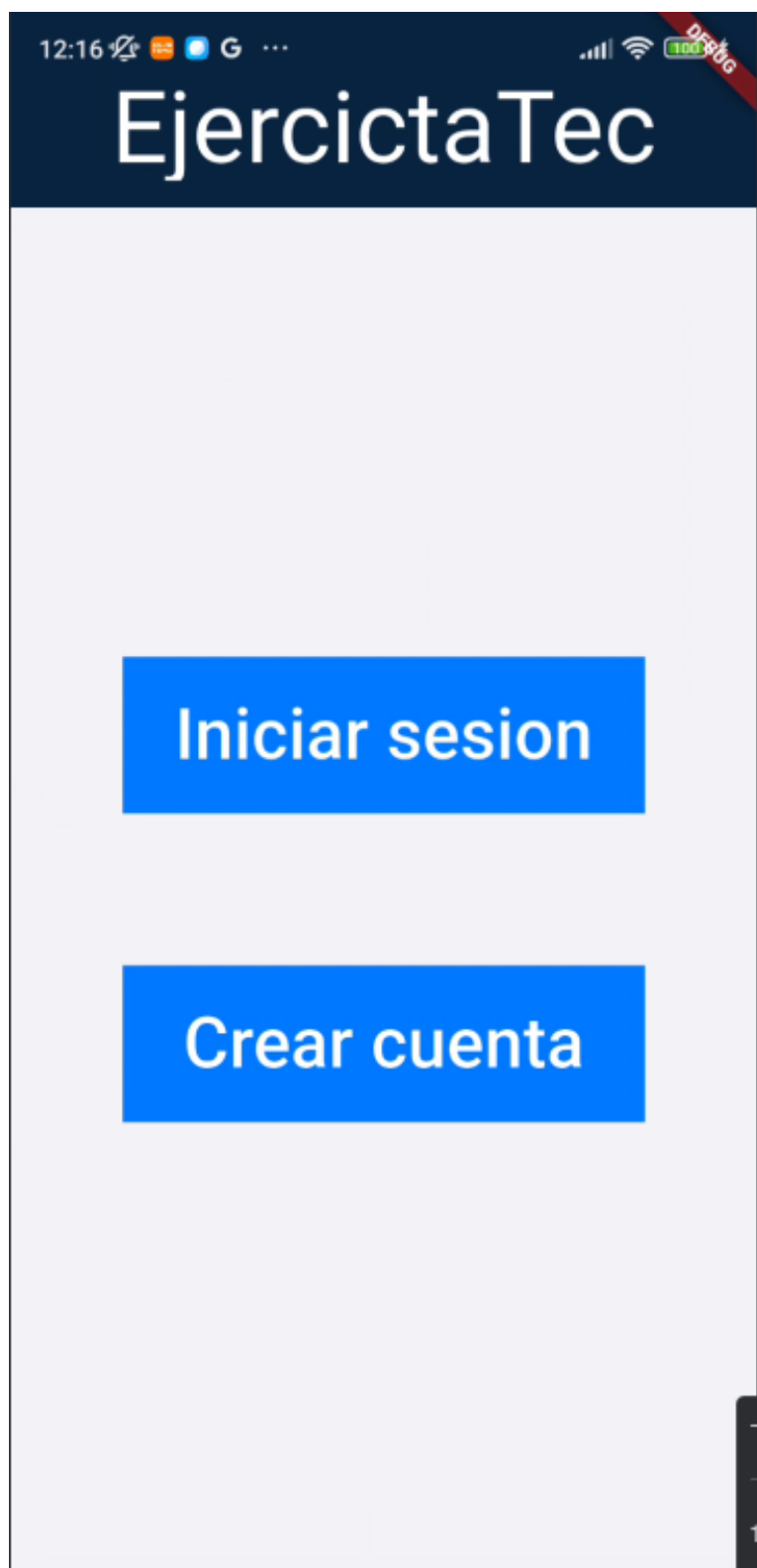
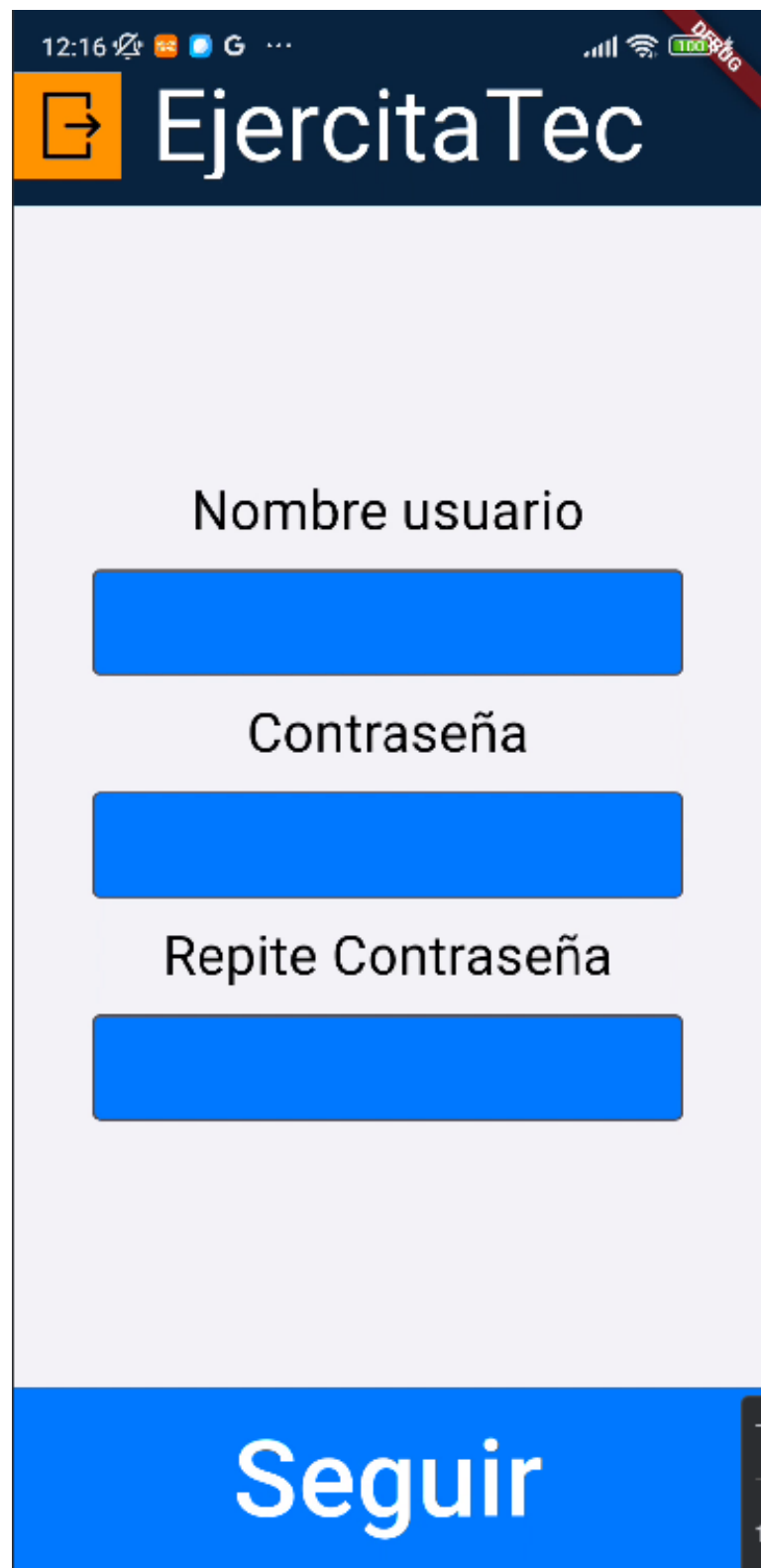



Figura 3.59: Log In / Sing In



12:16 100% 4G

 EjercitaTec

Nombre usuario

Contraseña

Repite Contraseña

Seguir

Figura 3.60: Sing In pantalla 1

12:16 100% 100%

EjercitaTec

Fecha Nacimiento

Seleccionar No seleccionada

Género

Prefiero no decirlo ▼

Peso en KG

Altura en cm


Crear usuario

Figura 3.61: Sing In pantalla 2

Las pruebas se realizaron en el backend, la correcta inserción y borrado de usuarios. Cuando se crea correctamente un usuario se redirige a la pantalla inicial. Si se intenta crear un usuario ya existente, se lanza un aviso. Pasó todas las pruebas.

SCRUM-20 Iniciar/Cerrar sesión
Tareas:
1. Implementar el boceto IU fig. 3.4 y parte necesaria de fig. 3.22
2. Implementar función de consulta en el backend
Pruebas de aceptación:
■ si el usuario con esa contraseña no existe, informar al usuario
■ si el usuario con esa contraseña existe, devolver un token para su guardado y guardar usuario
■ si el usuario quiere cerrar sesión, borrar token y nombre del usuario del dispositivo y volver a la pantalla inicial fig. 3.3

12:17 100% debug

 EjercictaTec

Nombre

Contraseña

Iniciar sesion

Figura 3.62: Log In

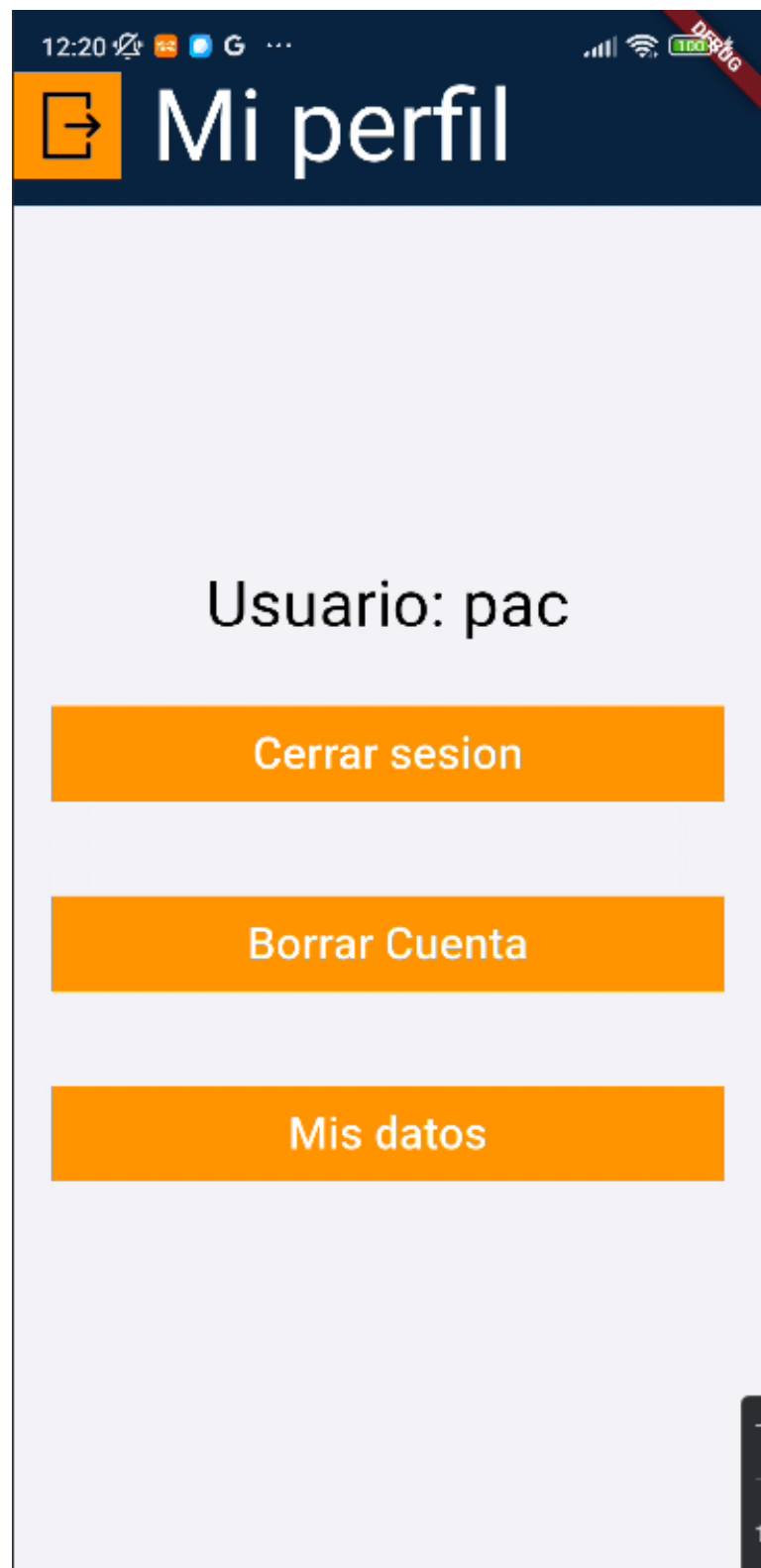


Figura 3.63: Opciones Usuario

Al hacer el login correctamente se otorga un token que no hay que renovar hasta el día siguiente, al cerrar sesión se borra automáticamente. Las pruebas realizadas se basaron en comprobar el correcto almacenamiento y borrado del token, usando la librería de secure storage de flutter, que nos permite guardar variables de forma permanente y encriptada con un fácil acceso, como si fuera un map.

Para comprobar la veracidad de las credenciales se implementó la parte básica del backend, un server con una BD(mySQL) que guardará los usuarios con sus contraseñas y la parte implementada usando Express.js(Node.js) para escuchar peticiones, el server también será el encargado de proporcionar y verificar los tokens. La comunicación entre servidor y cliente (el dispositivo en el que está instalada la app) se usan peticiones HTTP, en futuro se puede migrar a HTTPS si es necesaria más seguridad.

Se hicieron pruebas abriendo y cerrando sesión comprando si existía el token en memoria e intentando acceder a la app usando un token caducado. Siempre se añadía/borraba el token cuando se iniciaba/cerraba sesión y no dejaba entrar usando tokens caducados. Pasó todas las pruebas.

SCRUM-4 Buscar rutina en la lista del usuario
Tareas:
1. Implementar IU del boceto fig. 3.14
2. Implementar lo necesario en la BD local
3. Implementar algoritmo de búsqueda
4. Implementar acceso a la rutina seleccionada
Pruebas de aceptación:
■ si no existe la rutina buscada, no enseñar acceso en pantalla
■ si existe la rutina buscada, enseñar acceso en pantalla

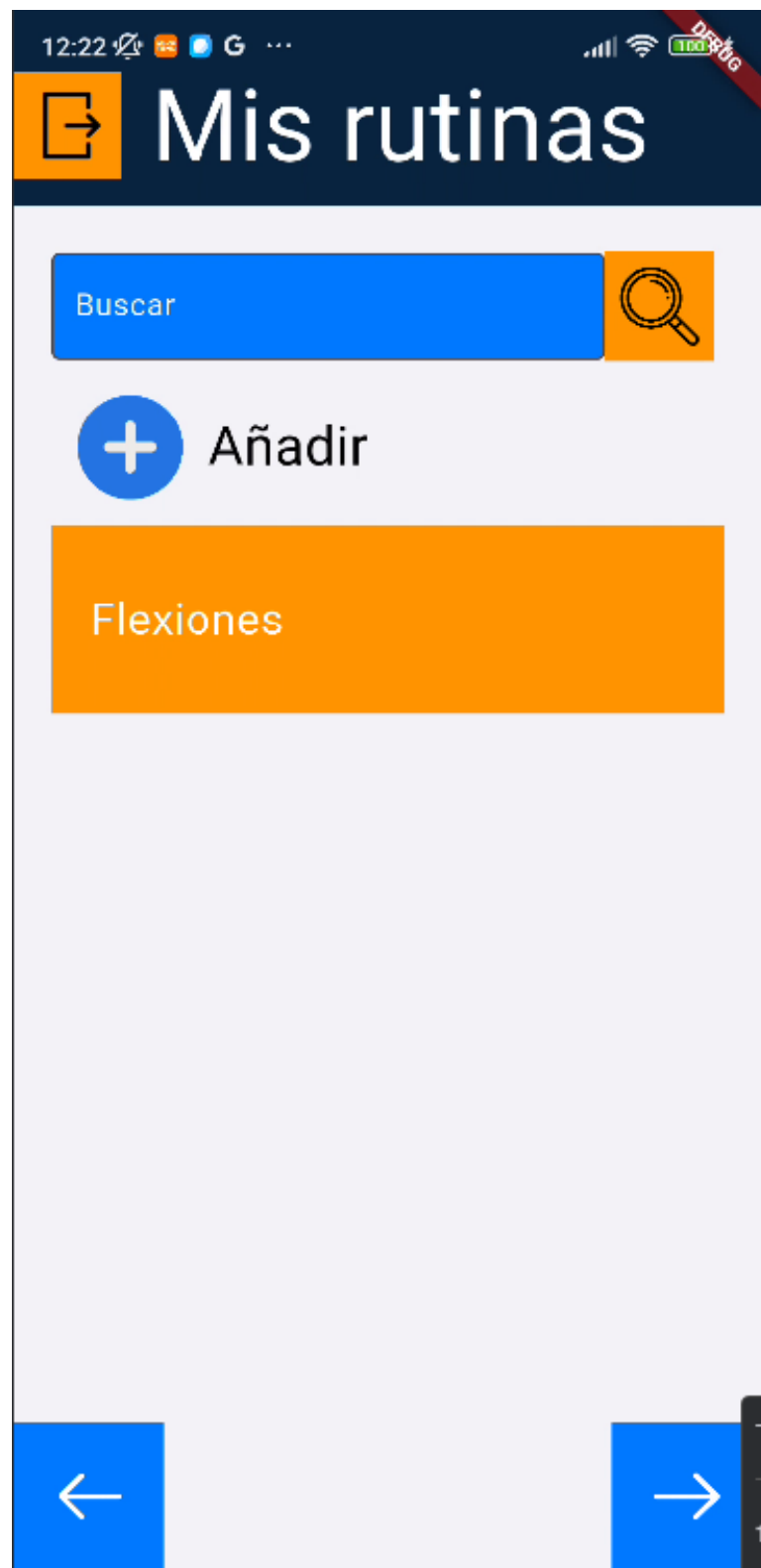


Figura 3.64: Lista Rutinas

Esta funcionalidad se abarcó aprovechando lo implementado en la iteración anterior del SCRUM-23. Se realizaron las mismas pruebas pero relacionadas con las rutinas. Como la funcionalidad previamente mencionada se diseñó para poder usar código a modo de plantilla y pasó todas las pruebas, esta como era de esperar las pasó también.

SCRUM-5 Insertar/Borrar/Modificar rutina
Tareas: 1. Implementar las insercciones en para la BD local 2. Implementar los borrados para la BD local 3. Implementar las modificaciones para la BD local
Pruebas de aceptación: <ul style="list-style-type: none">■ si se hace una insercción, que el elemento insertado sea visible en la lista de ejercicios■ si se hace un borrado, que el elemento desaparezca de la lista■ si se hace una modificación, que se hagan visibles los cambios al momento■ si hay algún fallo en alguna de estas operaciones, informar al usuario con el motivo

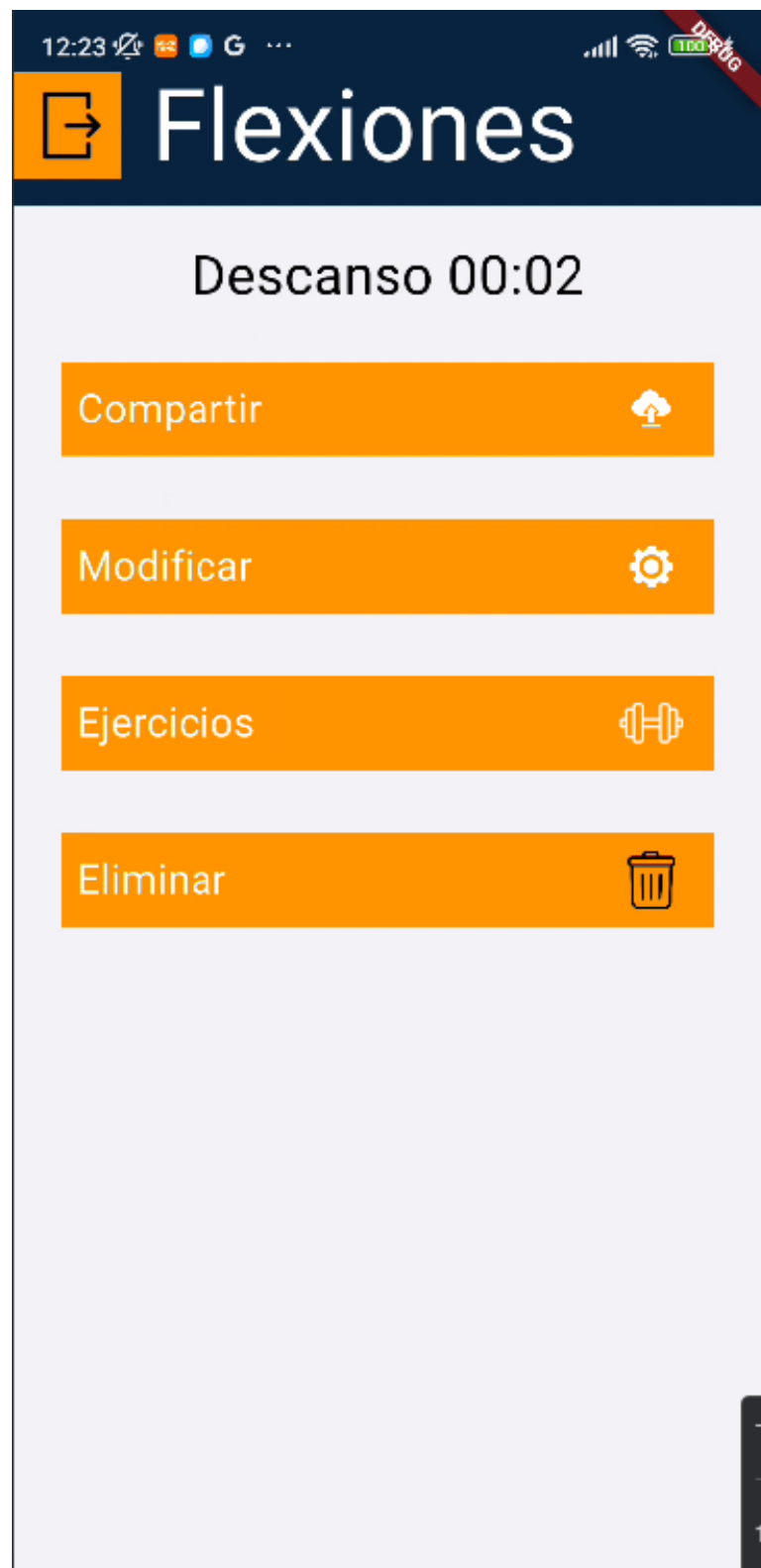


Figura 3.65: Datos Rutina

12:24 100%

Añadir rutina

Nombre

Descanso: mm:ss

Descripcion

Seguir

Figura 3.66: Crear Rutina

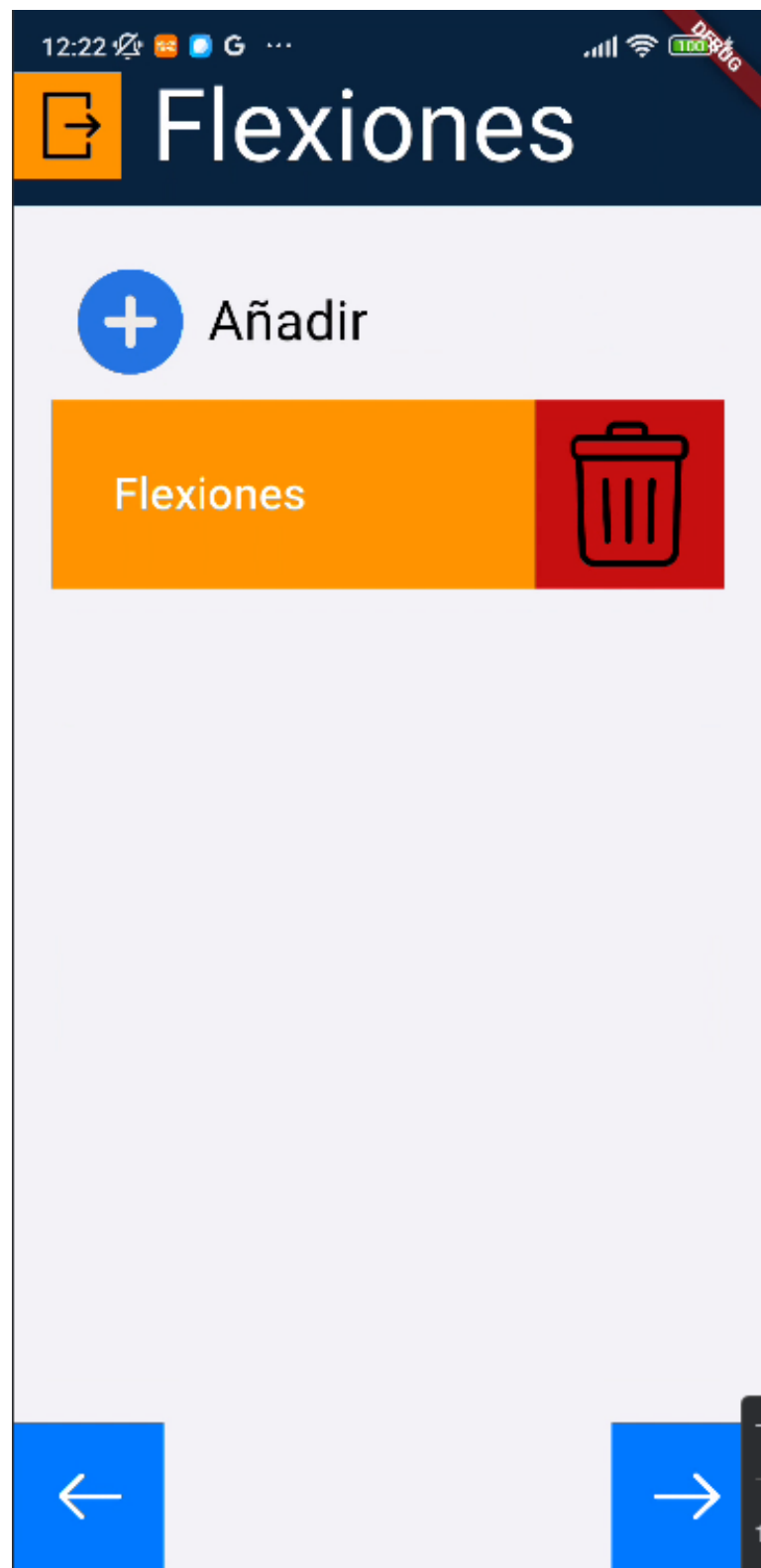


Figura 3.67: Lista Ejercicios Rutina

The screenshot shows a mobile application interface for editing a routine. At the top, there is a dark blue header with a white icon of a box with an arrow pointing right, followed by the title 'Flexiones' in large white text. Below the header, the form is divided into sections with light gray backgrounds. The first section is labeled 'Nombre' and contains a blue text input field with the text 'Flexiones'. The second section is labeled 'Descanso' and contains a blue text input field with the text '00:02'. The third section is labeled 'Descripcion' and contains a large blue text input field with the text 'Entrenamiento'. At the bottom of the form, there is an orange button with the text 'Guardar' and a white save icon. The status bar at the top of the phone shows the time 12:22, signal strength, Wi-Fi, and battery level at 100%.

12:22

Flexiones

Nombre

Flexiones

Descanso

00:02

Descripcion

Entrenamiento

Guardar

Figura 3.68: Modificar Rutina

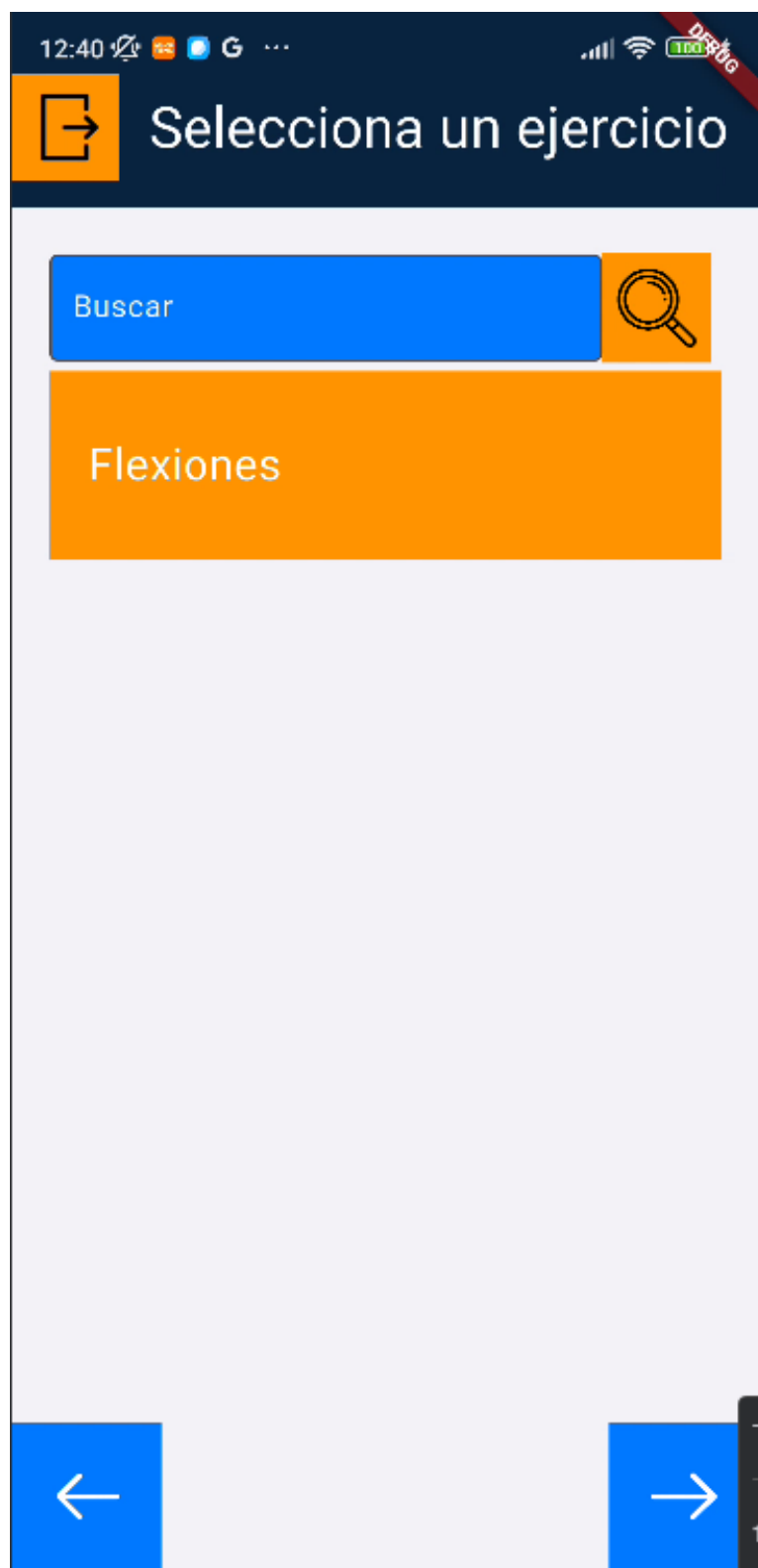


Figura 3.69: Lista añadir ejercicio a rutina

Como se puede observar se rescato mucho código de la primera iteración, solo hubo que hacer unas pequeñas modificaciones en la clase plantilla usada para crear las pantallas fig. 3.67 y fig. 3.69, en concreto a la pantalla de la lista de ejercicios perteneciente a la rutina se le realizaron pruebas para ver la correcta actualización de los contenidos.

En la pantalla fig. 3.68 la única prueba realizada fue comprobar la correcta modificación en la BD local y su correcta visualización en la misma.

La funcionalidad de compartir todavía no está implementada.

3.9.3.1. Sprint Review 2

Se sugirieron mejoras menores como el ajuste de tamaños e iconos en los botones de la sección de creación de rutinas.

También se modificó la funcionalidad de compartir rutinas. Ahora todas son modificables, eliminando la distinción entre rutinas descargadas (antes no modificables) y creadas (modificables). Esto mejora la experiencia del usuario: si desea volver a una rutina original, simplemente la puede volver a buscar y descargar.

Se cumplieron todas las funcionalidades en este sprint.

3.9.3.2. Estado de la BD local

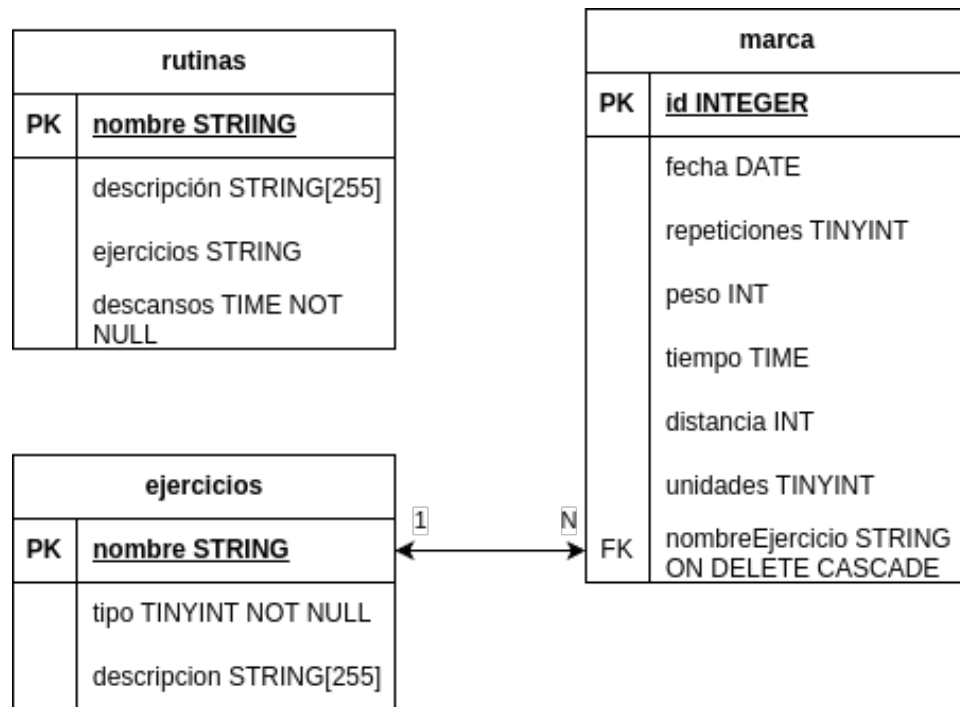


Figura 3.70: BD local iteracion 2

Se añadieron las tablas de ejercicios y marcas, una o varias marcas siempre están relacionadas con un ejercicio, de esta forma, se saben todas las marcas de un ejercicio y si se elimina un ejercicio sus marcas también desaparecen.

3.9.3.3. Estado de la BD backend

Antes de seguir, la BD del backend es donde se guardan los usuarios registrados y sus rutinas compartidas. Para diferenciarlo de la BD local, que es el almacenamiento interno del dispositivo.

usuario	
PK	<u>username VARCHAR[50]</u>
	passwd VARCHAR[255]

Figura 3.71: BD backend iteracion 2

3.9.4. Iteración 3

Esta iteración se centró en desarrollar la funcionalidad para compartir rutinas entre usuarios, incluyendo la subida, visualización y descarga de rutinas. Además, se comenzó a implementar la funcionalidad de resumen de datos para optimizar el uso de la memoria local. Incluye las siguientes historias:

- SCRUM-2 Establecer peso objetivo
- SCRUM-29 Copiar rutina
- SCRUM-8 Enseñar datos de una rutina a descargar
- SCRUM-9 Compartir mi rutina
- SCRUM-27 Buscar rutinas para descargar
- SCRUM-7 Revisar datos para ver si el descanso es necesario
- SCRUM-19 Resumir datos

SCRUM-2 Establecer peso objetivo
Tareas:
1. Implementar la parte necesaria de fig. 3.22
2. Implementar el almacenaje de ese peso objetivo
Pruebas de aceptación:
<ul style="list-style-type: none"> ■ si el peso objetivo está en el formato correcto, guardar ■ si el peso objetivo no está en el formato correcto, enseñar una alerta para su corrección

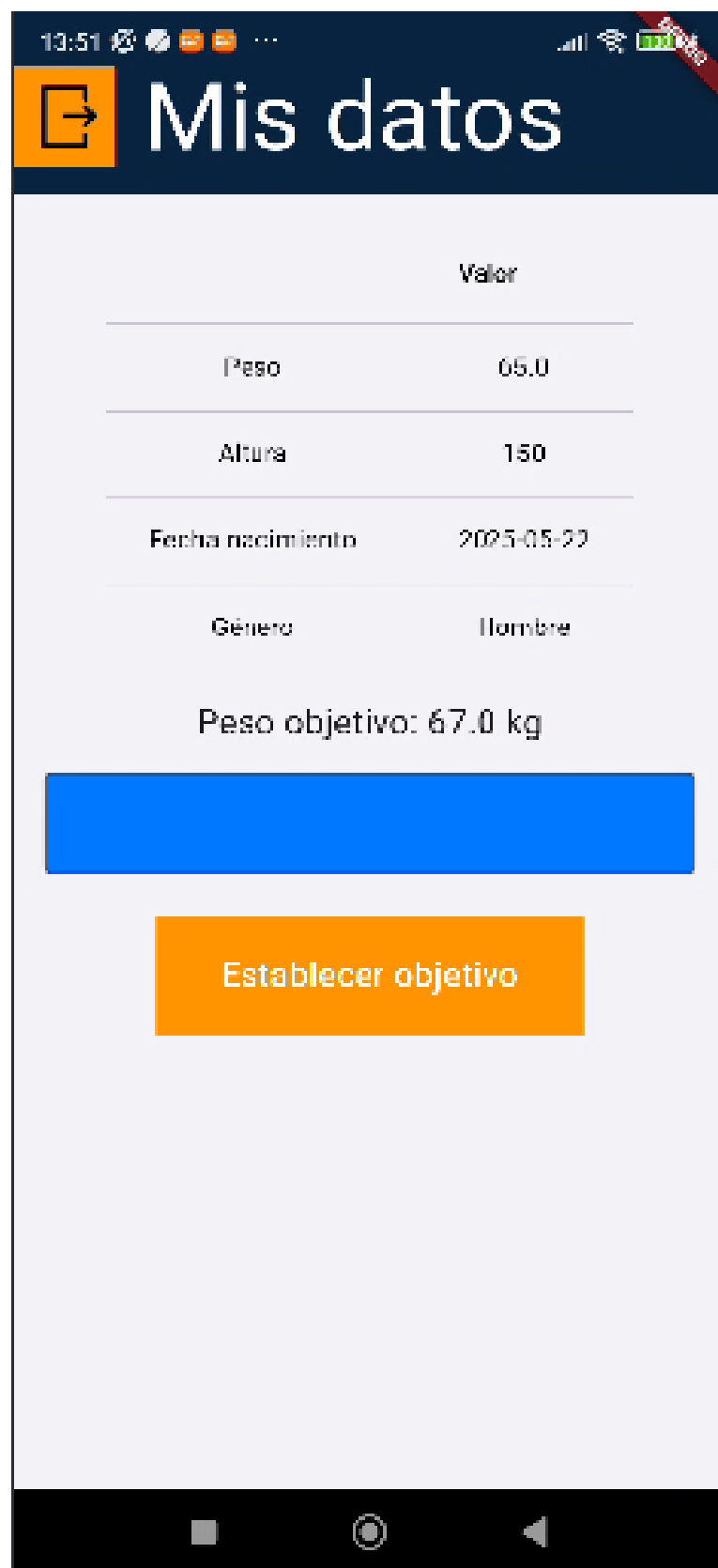


Figura 3.72: Peso Objetivo

Para guardar el peso, en esta iteración como no se crearan las tablas para guardas los pesajes del usuario se guardará usando secure storage, es una solución temporal. En para esta funcionalidad se hicieron pruebas para comprobar la correcta insercción del formato del peso, que no se añadan letras. que se admitan comas y puntos, que solo se use un decimal. Pasó todas las pruebas.

SCRUM-9 Compartir mi rutina
Tareas:
1. Implementar la BD en el backend
2. Implementar funciones de inserción en la API
Pruebas de aceptación:
■ si la rutina no contiene ejercicios, no permitir su subida al servidor
■ si la rutina contiene ejercicios, permitir la subida al servidor

Esta funcionalidad dió muchos quebraderos de cabeza ya que varios usuarios podrían tener una rutina con el mismo nombre y al mismo tiempo al descargar una rutina, un usuario podría tener un ejercicio llamado igual o varios usuario podrían tener en la nube varios ejercicios subidos con el mismo nombre. Esto son solo uno de los pocos problemas encontrados en esta funcionalidad. Haciendo pruebas de esta funcionalidad sobre integridad usando distintos casos se hizo conocer lo mencionado previamente.

Para solucionar algunos de estos problemas se hizo lo siguiente, en el backend en la BD los ejercicios y rutinas serán guardados usando como clave primaria un id que sera un int autoincrementado, evitando problemas integridad. También cuando sean referenciados en la BD se hará usando su id.

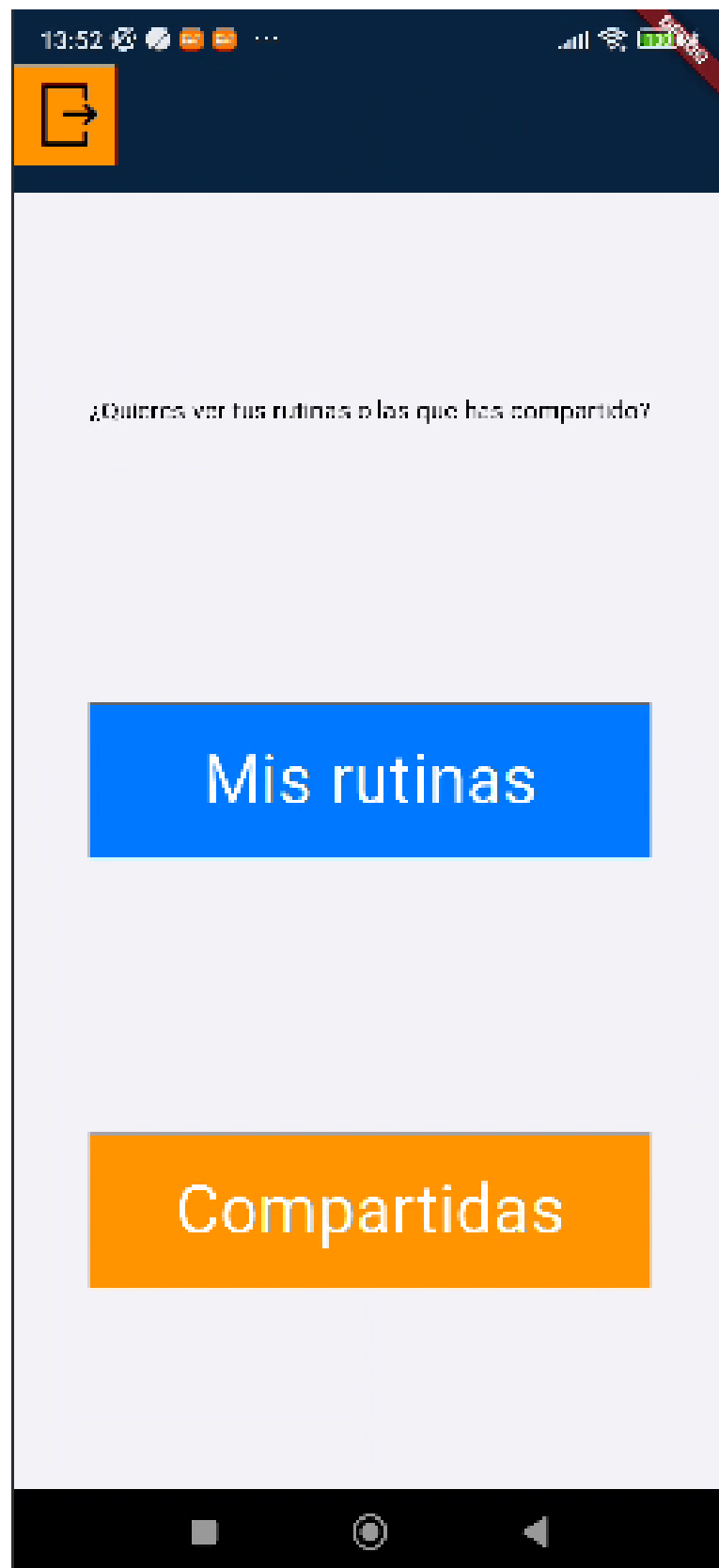


Figura 3.73: Elegir lista compartida por el usuario o locales

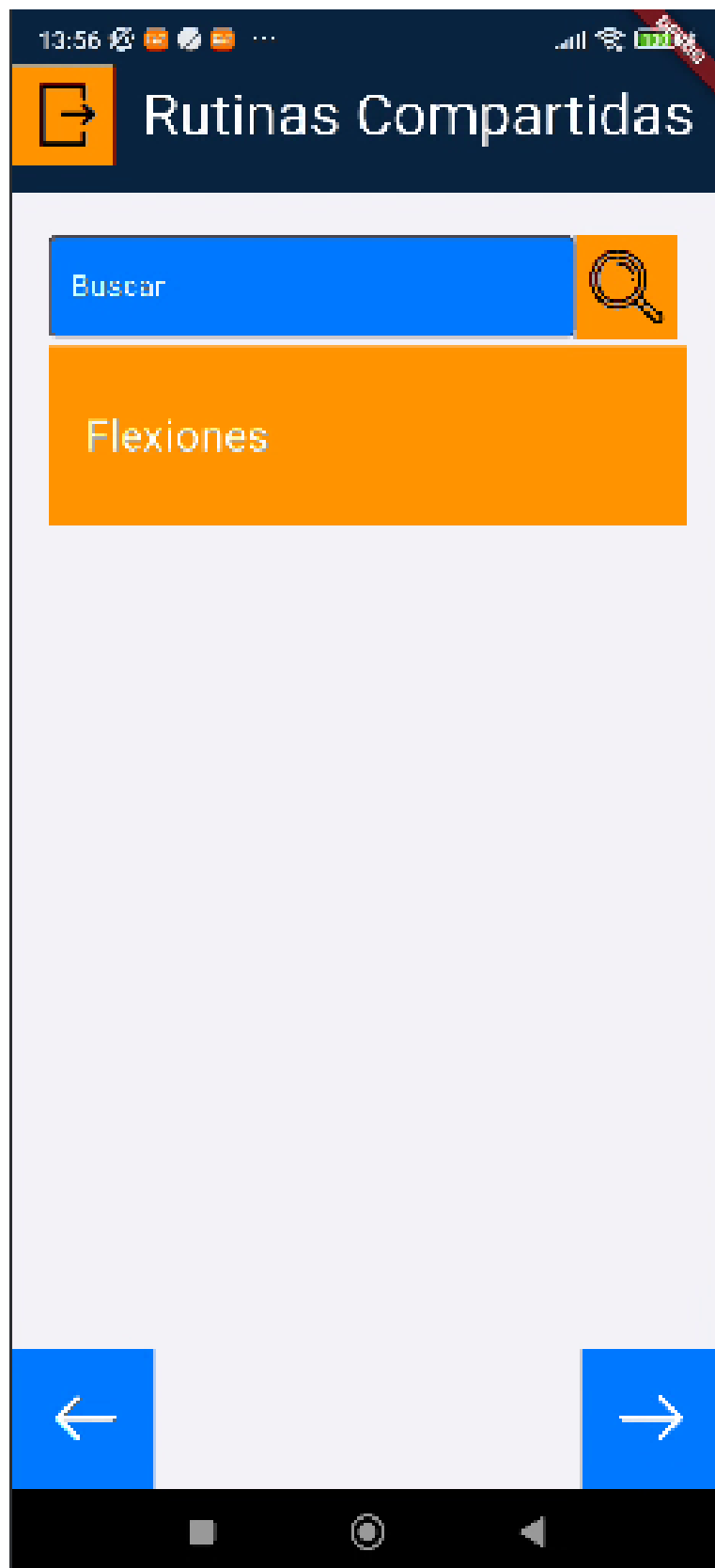


Figura 3.74: Rutinas compartidas por el usuario

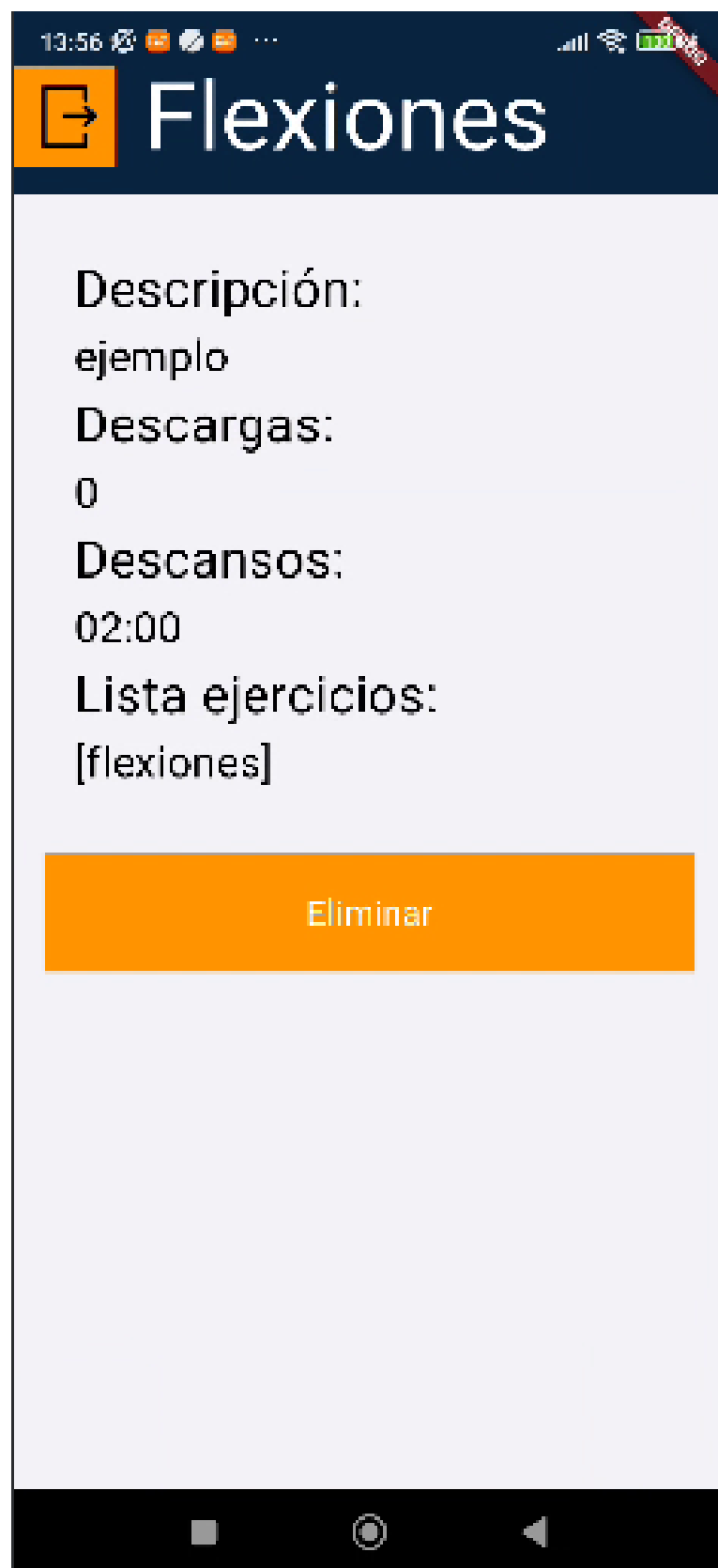


Figura 3.75: Datos rutinas compartidas por el usuario

SCRUM-27 Buscar rutinas para descargar
<p>Tareas:</p> <ol style="list-style-type: none"> 1. Implementar funciones de consultas en la API 2. Implementar IU de búsqueda de rutinas por usuario creador y por nombre de rutina <p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> ■ si busco un usuario que no existe, no se muestra ningún acceso en la IU ■ si busco un usuario existente, se muestra su acceso en la IU ■ si busco una rutina que no existe, no se muestra ningún acceso en la IU ■ si busco una rutina existente, se muestra su acceso en la IU

Se puede buscar tanto un usuario y ver las rutinas que ha subido o directamente buscar por nombre de la rutina. Se recicló la pantalla de lista de rutinas fig. 3.14, y se implementó la pantalla de DatosRutinaComp, que se usa para ver datos de pantallas compartidas. También para hacer más fácil la distinción entre rutinas del propio usuario y las que ha compartido se añadió otra pantalla exclusivamente de pantallas compartidas, en la cual se nos permite eliminar de la nube las rutinas elegidas.

Haciendo pruebas descargando varias veces la misma rutina y casos distintos fue fácil detectar el motivo de los problemas. Para solucionar los problemas de integridad generados en el dispositivo del usuario, al descargar una rutina se le añadirá a su nombre y al de los ejercicios que contiene el nombre del creador, y si aún así hay problemas de integridad, se añade al nombre de la rutina y/o ejercicio el número de copia que es dentro del dispositivo. Desarrollando lo anterior, al descargar una rutina llamada "Flexiones" del usuario "Paco", en el dispositivo que realiza la descarga se llamará "Flexiones-Paco", si en el dispositivo en el que se realiza la descarga ya existe otra rutina llamada "Flexiones-Paco", la descargada se llamará en el dispositivo "Flexiones-Paco(1)". La integridad de los ejercicios se trata igual.

Una vez implementadas estas soluciones, ya se pasaron exitosamente las pruebas.

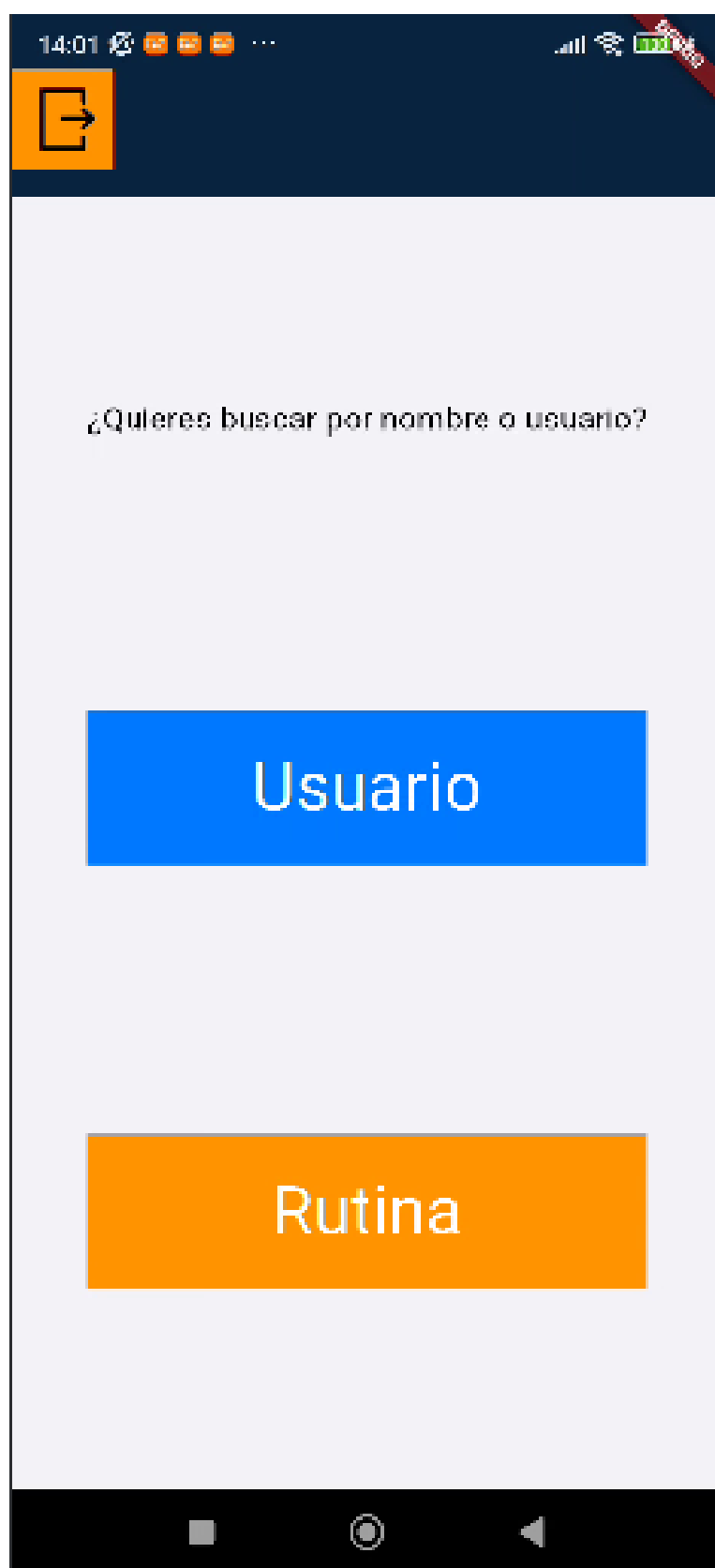


Figura 3.76: Buscar rutinas o usuario

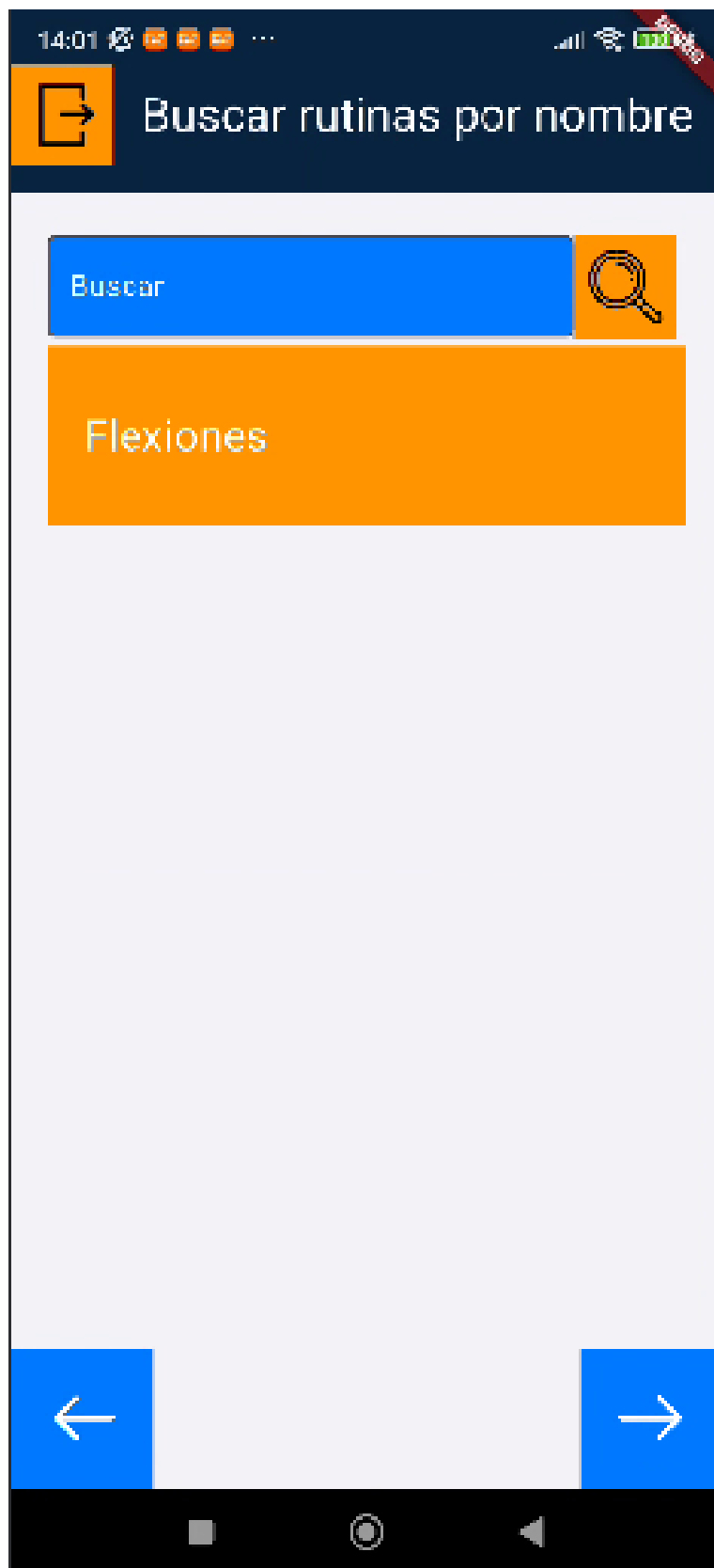


Figura 3.77: Buscar rutinas por su nombre

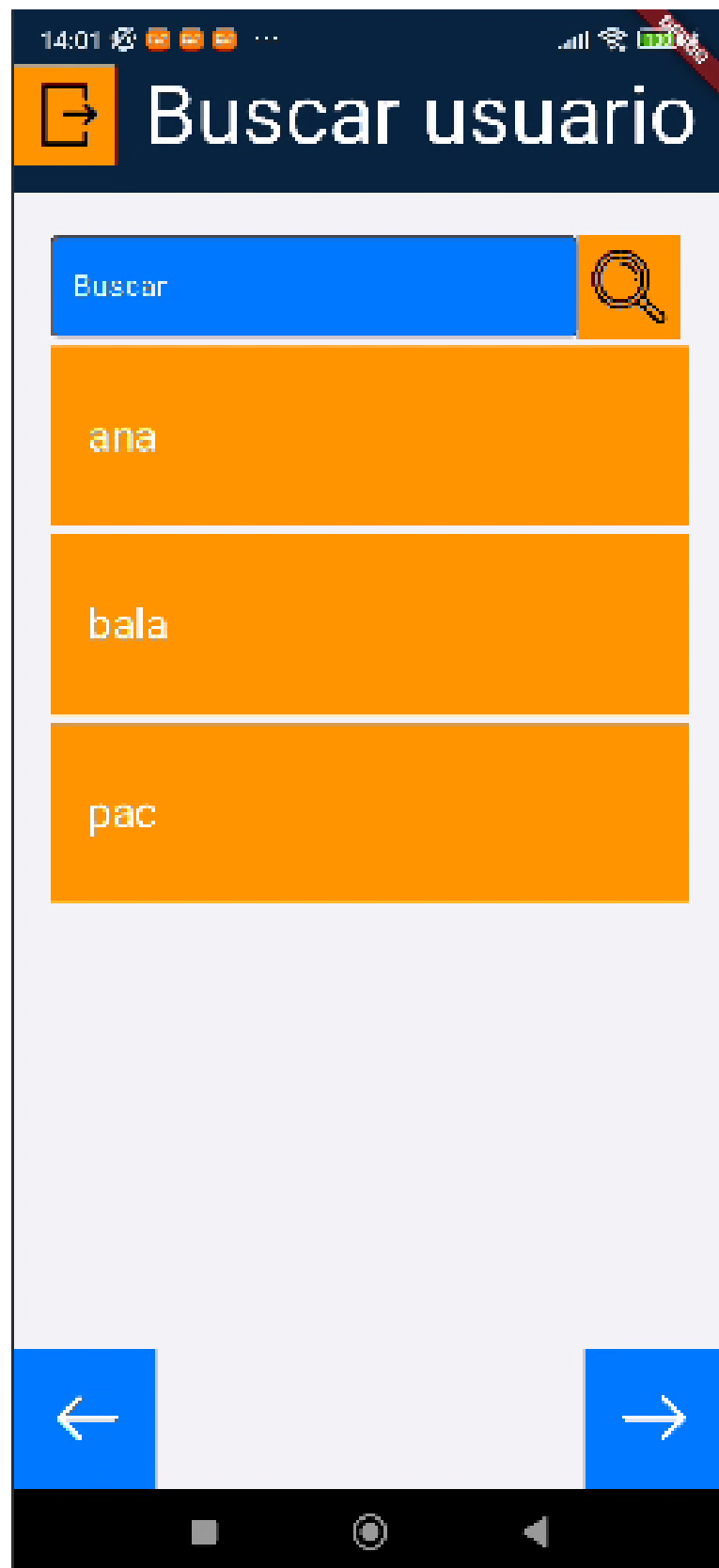


Figura 3.78: Buscar usuarios

SCRUM-8 Enseñar datos de una rutina a descargar
Tareas: <ol style="list-style-type: none">1. Implementar el IU pop up2. Implementar las funciones de consulta en la API3. Implementar las funciones de inserción en la BD local
Pruebas de aceptación: <ul style="list-style-type: none">■ si se va a descargar la rutina y el usuario tiene en el dispositivo otra con el mismo nombre, un algoritmo resolverá este problema■ si se va a descargar la rutina y el usuario no tiene en el dispositivo otra con el mismo nombre, descargar normalmente

Es la misma pantalla para ver nuestras rutinas subidas, reciclamos código, la única diferencia es que no podemos eliminarla de la nube, solo podemos descargarla. Se hicieron pruebas comprobando que solo podía borrar la rutina el usuario creador y que los datos enseñados al descargar la rutina eran correctos.

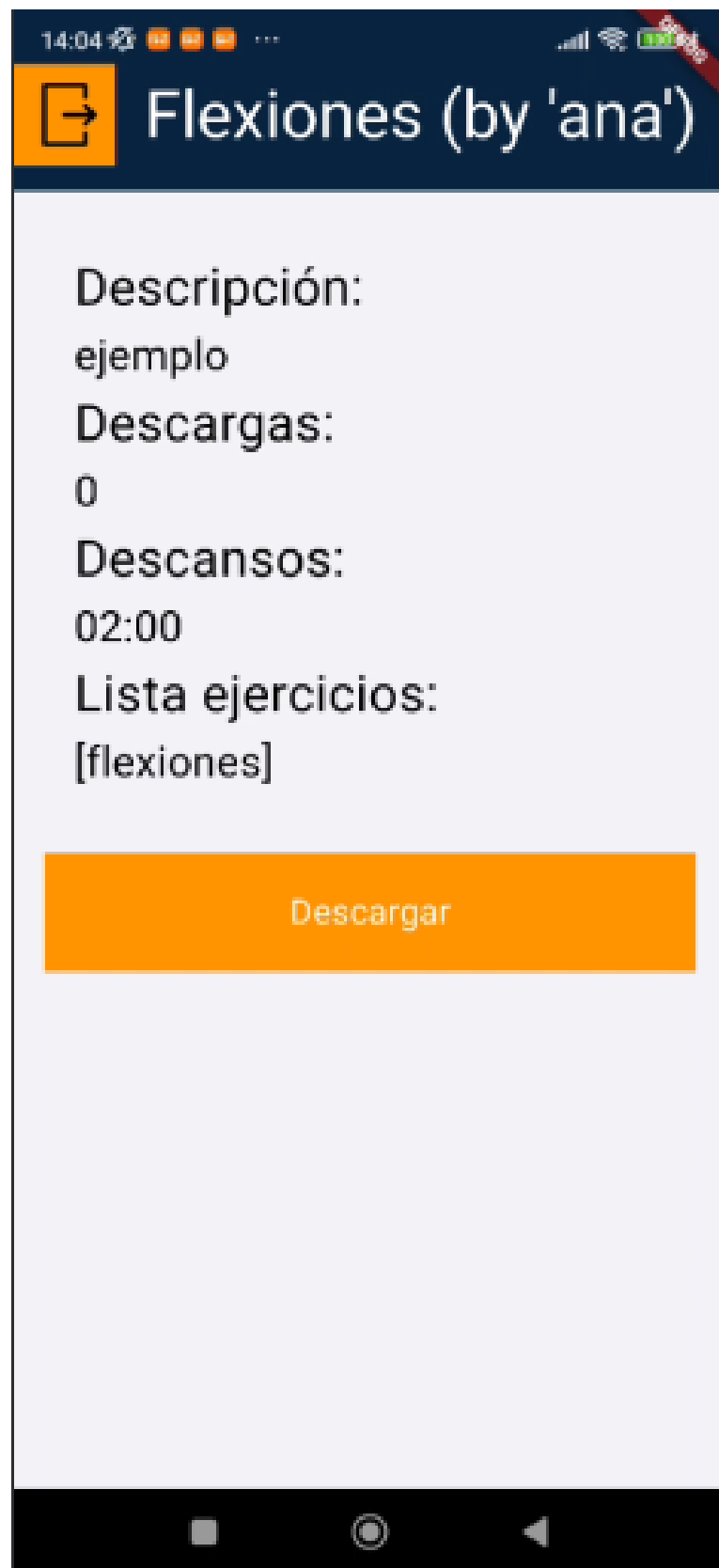


Figura 3.79: Datos rutinas para descargar

SCRUM-29 Copiar rutina
Tareas: <ol style="list-style-type: none">1. Añadir al boceto IU de datos rutina fig. 3.16 esta funcionalidad2. Implementar la IU resultante de la tarea anterior3. Implementar las consultas e inserciones necesarias en la BD local
Pruebas de aceptación: <ul style="list-style-type: none">■ si hay algún problema por coincidencia de nombres, un algoritmo resolverá este problema■ si no hay algún problema por coincidencia de nombres, copiar

3.9.4.1. Funcionalidad descartada

Durante la realización de estas tareas de usuario, se dió notoriedad a que las historias restantes de este sprint (SCRUM-7 Revisar datos para ver si el descanso es necesario y SCRUM-19 Resumir datos) no pueden ser implementadas en este punto del desarrollo. Se pospondrán para su futura implementación.

Se ha decidido dar menos prioridad y que por tanto no se abordará en este TFG por limitaciones de tiempo en concreto el SCRUM-7 (Revisar datos para ver si el descanso es necesario), porque necesitaría los datos de un smartwatch y ha sido una funcionalidad previamente descartada.

También se ha reducido la prioridad de SCRUM-29 (Copiar rutina), dado que aporta poco valor a la app.

Se decidió adelantar otras, pero debido a los problemas encontrados durante esta iteración no se pudieron realizar.

3.9.4.2. Estado BD local

No ha habido cambios respecto a la iteración anterior

3.9.4.3. Estado BD backend

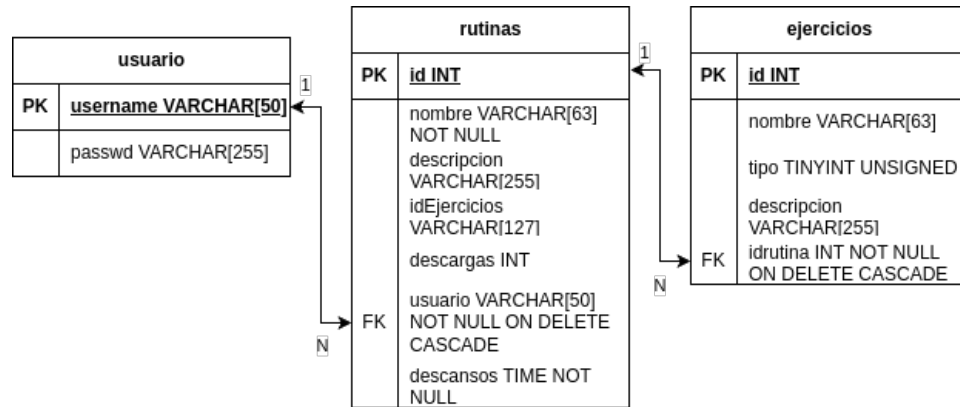


Figura 3.80: BD backend iteracion 3

3.9.4.4. Sprint Review 3

Se identificaron dos errores principales de planificación:

Error 1: Se planificó implementar la funcionalidad de resumir datos sin haber completado la obtención de datos.

Solución: Replanificar los sprints.

Los imprevistos y problemas surgidos durante el desarrollo han permitido detectar *bugs*, errores de diseño y carencias funcionales que de otro modo podrían haber pasado desapercibidos. Gracias a ello, se han podido proponer nuevas funcionalidades y mejorar las ya existentes, lo cual contribuye significativamente a aumentar la calidad global de la aplicación.

Algunas funcionalidades propuestas como mejora son:

- Verificar el token del usuario antes de permitir la subida de rutinas, para evitar suplantación.
- Al seleccionar la opción de borrar usuario del dispositivo, eliminar también la base de datos local asociada.
- Crear una base de datos local independiente para cada nuevo usuario creado en un dispositivo.
- Posibilidad de editar el nombre de una rutina.

- Marcar los ejercicios eliminados con una *flag*, para evitar que se inicien rutinas que los contengan.
- Marcar a los usuarios permanentemente eliminados con una *flag*, para proceder a eliminarlos en los dispositivos.

Estas funcionalidades están pensadas para aportar mayor **seguridad, integridad y calidad** al producto final.

En esta review se estuvo debatiendo si quitar las funcionalidades relacionadas con el smartwatch, por cuestiones de tiempo, ya que con las funcionalidades añadidas en el capítulo anterior se va ajustando el plazo.

En esta iteración se tenía pensado implementar la funcionalidad de resumir los datos, no obstante, al no tener implementada la parte en la que guardo las marcas de los entrenamientos, no puedo trabajar en esa funcionalidad. Por lo tanto, se ha pospuesto para la próxima iteración. También se le dió el visto bueno a la implementación de las funcionalidades previamente expuestas durante el desarrollo.

3.9.5. Iteración 4

En esta iteración se tiene pensado implementar la funcionalidad del calendario, donde el usuario podrá ver su planificación y acceder a sus marcas. Por otro lado, también se implementarán las funcionalidades nuevas del sprint anterior y las nuevas pantallas.

- SCRUM-26: Detectar metas cumplidas en los ejercicios después del entrenamiento
- SCRUM-1: Registrar peso por día
- SCRUM-16: Realizar el flujo del entrenamiento
- SCRUM-28: Implementar calendario

SCRUM-28: Implementar calendario
<p>Tareas:</p> <ol style="list-style-type: none">1. Implementar la IU e insertarla en el menu principal fig. 3.72. Implementar en la BD local3. Implementar funciones de inserción en BD local4. Implementar funciones de consulta en BD local5. Implementar funciones de borrado en BD local6. Implementar un acceso para ver los datos registrados sobre un día en concreto
<p>Pruebas de aceptación:</p> <ul style="list-style-type: none">■ si un día no tiene una rutina asignada, se considera descanso■ si un día tiene una rutina asignada, no hay marcas registradas y el día ya ha pasado, no se enseñará información, ya que se considera no entrenado■ si un día tiene una rutina asignada, no hay marcas registradas pero es hoy, se enseñará un acceso para empezar el entrenamiento, se considera que el usuario todavía no ha entrenado■ si un día tiene una rutina asignada, pero es futuro, no se enseñará un acceso para empezar el entrenamiento■ si un día tiene una rutina asignada y hay marcas registradas sea el día actual o pasado, se mostrará información de las marcas, sin acceso a entrenamiento

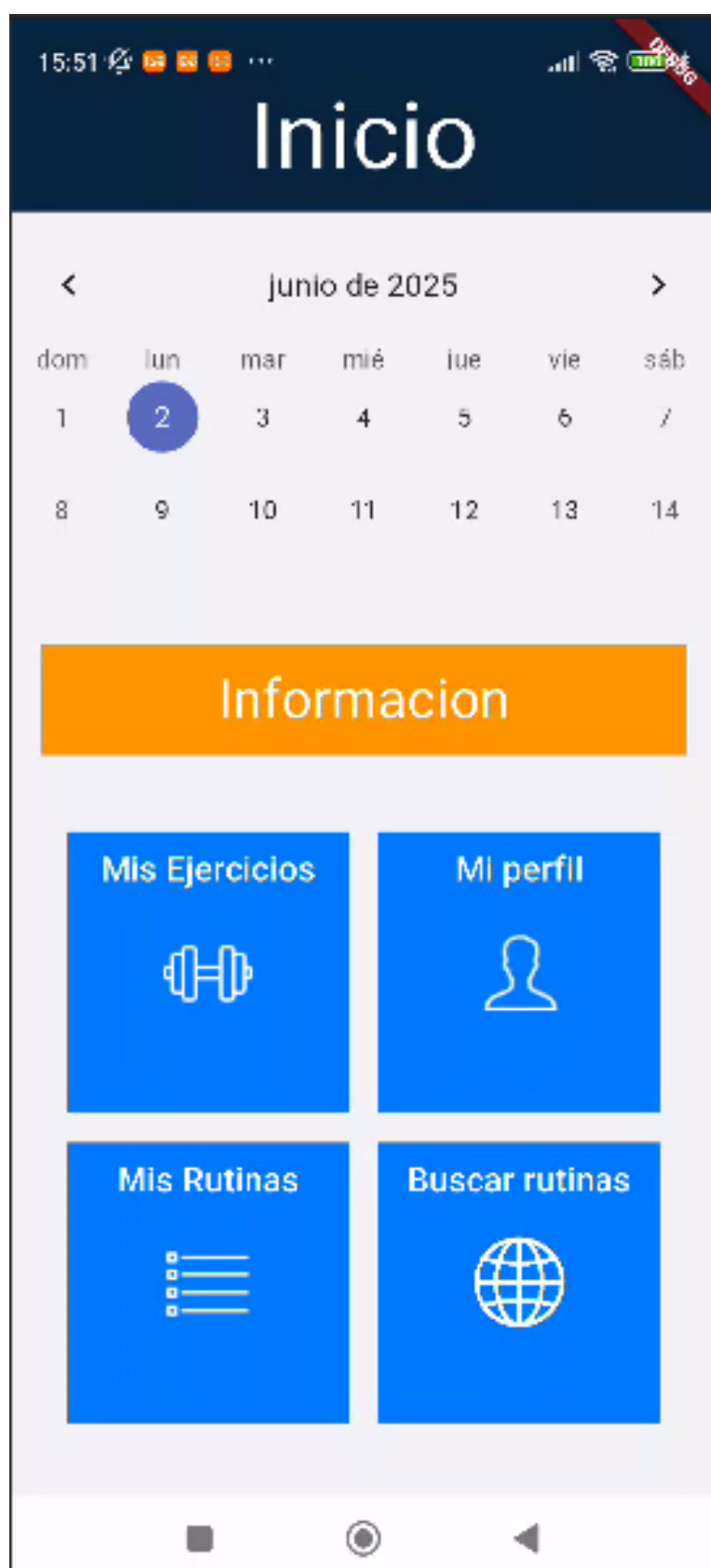


Figura 3.81: Calendario

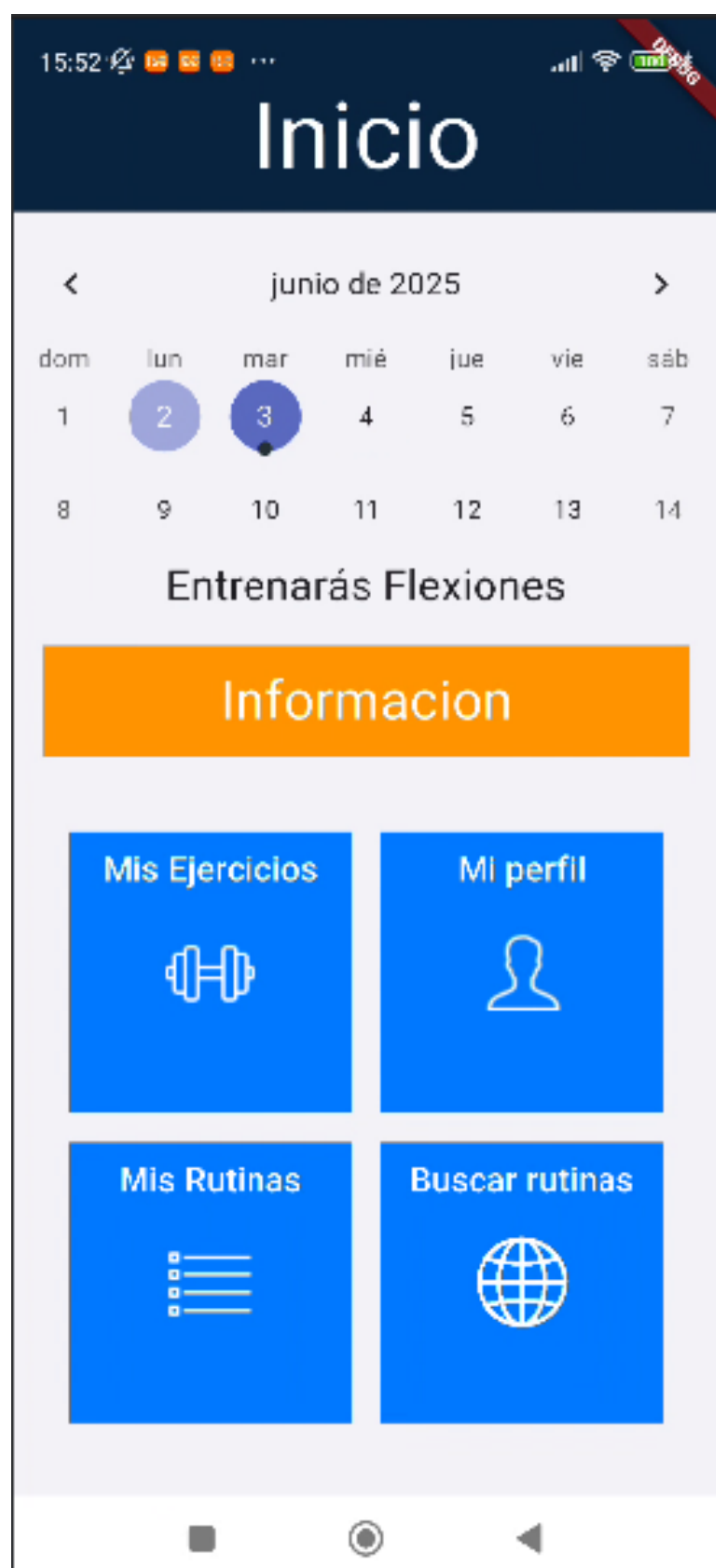


Figura 3.82: Calendario con eventos

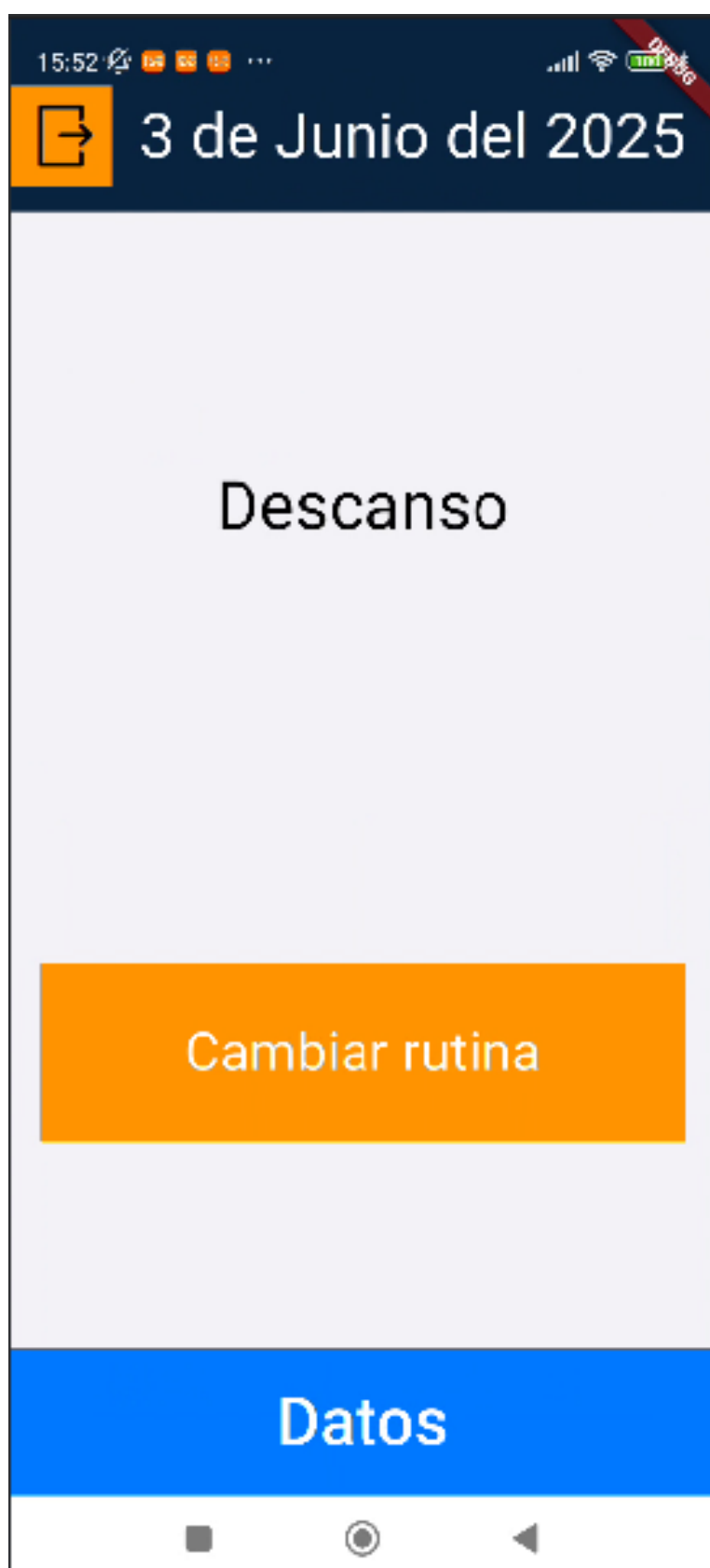


Figura 3.83: Descanso

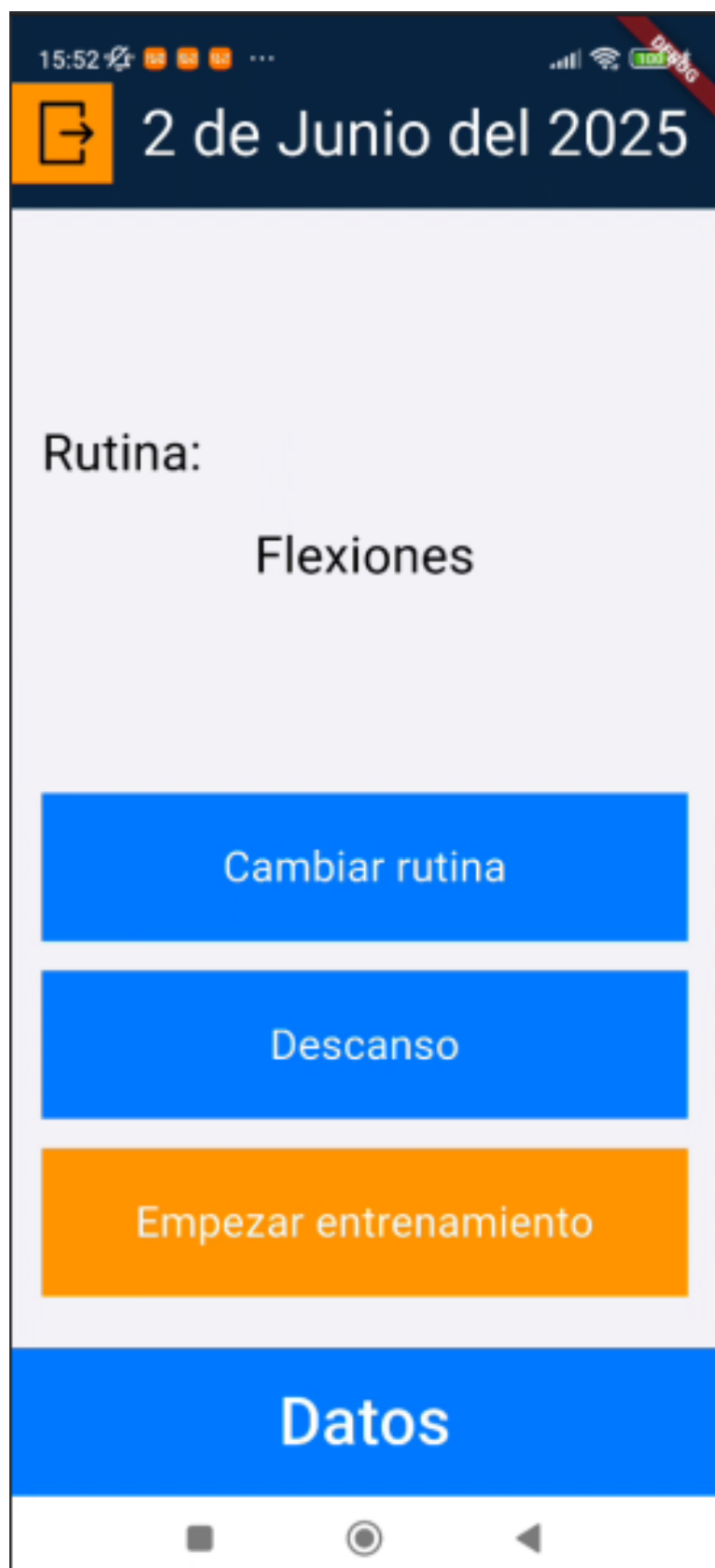


Figura 3.84: Entrenar

El calendario es donde el usuario podrá organizar sus entrenamientos, acceder a sus marcas y datos sobre su peso guardados por días. El calendario está implementado usando la librería de tableCalendar, que permite trabajar con calendarios con una alta personalización de la estética y trabajar comodamente con los datos reflejados en el mismo usando el tipo DateTime de flutter. Para guardar eventos los almaceno en una tabla en la BD local llamada entrenamientos. Cada entrenamiento se guarda usando la fecha de cuando se realizó/realizará como primary key junto con la rutina que se hizo o planea hacerse. Como no dejo modificar las rutinas una vez ya se haya entrenado o pasado el día, el único problema de integridad es cuando añado un evento a un día ya asignado, lo resuelvo sustituyendo el nuevo por el antiguo. Las marcas funcionan de manera similar, primary key es un int auto incrementado, pero tiene asociado cada marca una fecha. Importante, cada marca se asocia a los resultados de una serie.

Se realizaron pruebas comprobando si se actualizaban los entrenamientos/-pesajes añadidos se veían reflejados correctamente. Todos los datos se guardan correctamente así que pasó las pruebas, el único fallo es que a veces no actualiza la IU correctamente, por lo que se dijo previamente de que no se puede actualizar un widget dentro de otro, pero solo es visual.

SCRUM-16 Realizar el flujo del entrenamiento
<p>Tareas:</p> <ol style="list-style-type: none"> 1. Implementar IU de las pantallas de entrenamientos(de la fig. 3.34 a la fig. 3.41) 2. Implementar una clase que controle el cambio entre distintas pantallas y el guardado de marcas
<p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> ■ si la rutina asignada para un día, no tiene ejercicios y se desea iniciar el entrenamiento, se muestra por pantalla una alerta y no se permite continuar con el entrenamiento ■ si la rutina asignada para un día, tiene ejercicios, comienza el recorrido entre IUs ■ si es la primera serie de un ejercicio, no se permite terminar el ejercicio ■ si no es la primera serie de un ejercicio, se permite terminar el ejercicio ■ si acaba una serie, no se permitirá acabar el ejercicio o continuar con este hasta que termine el tiempo de descanso que se muestra en pantalla ■ si finaliza el último ejercicio de la lista, fin del entrenamiento, se devuelve al usuario al menú principal y guardan las marcas obtenidas

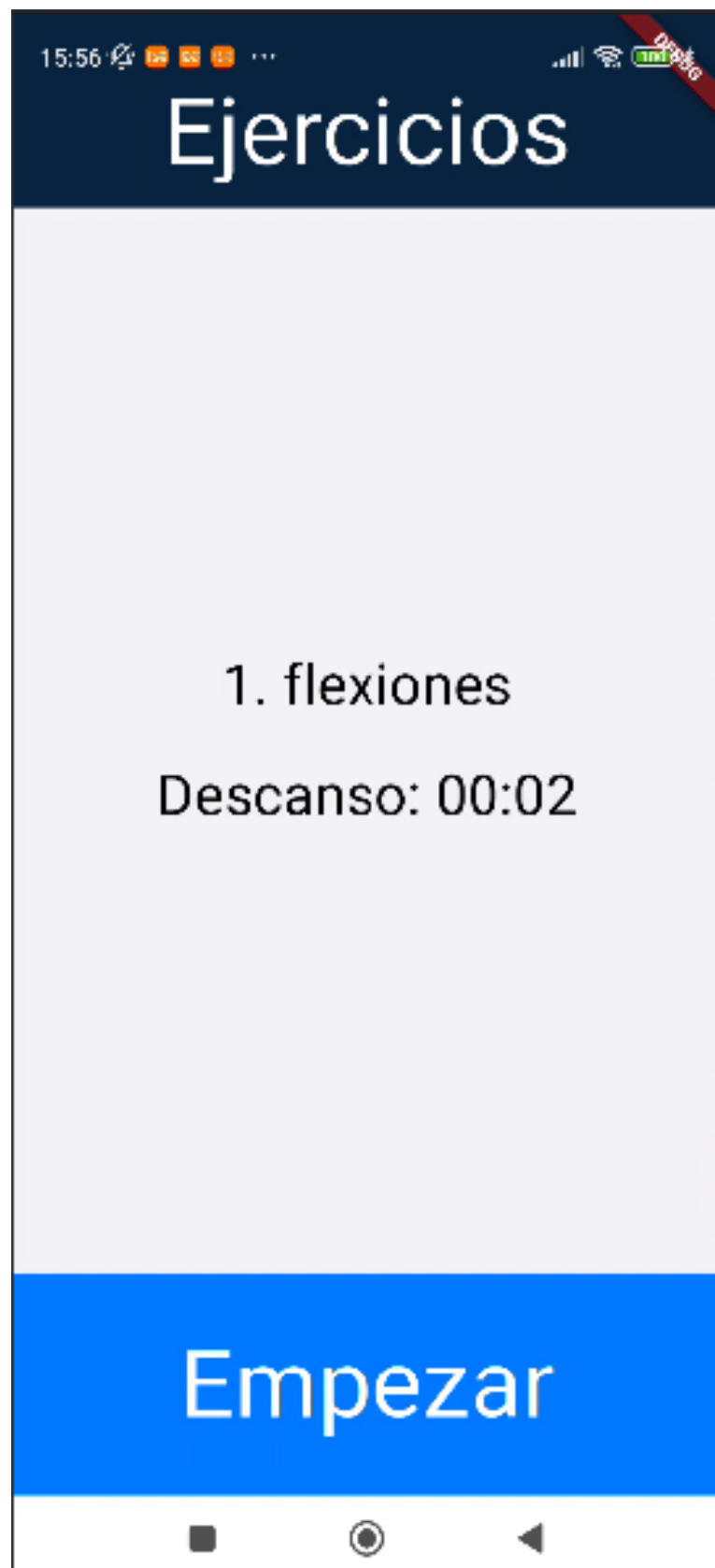


Figura 3.85: Lista inicial de ejercicios

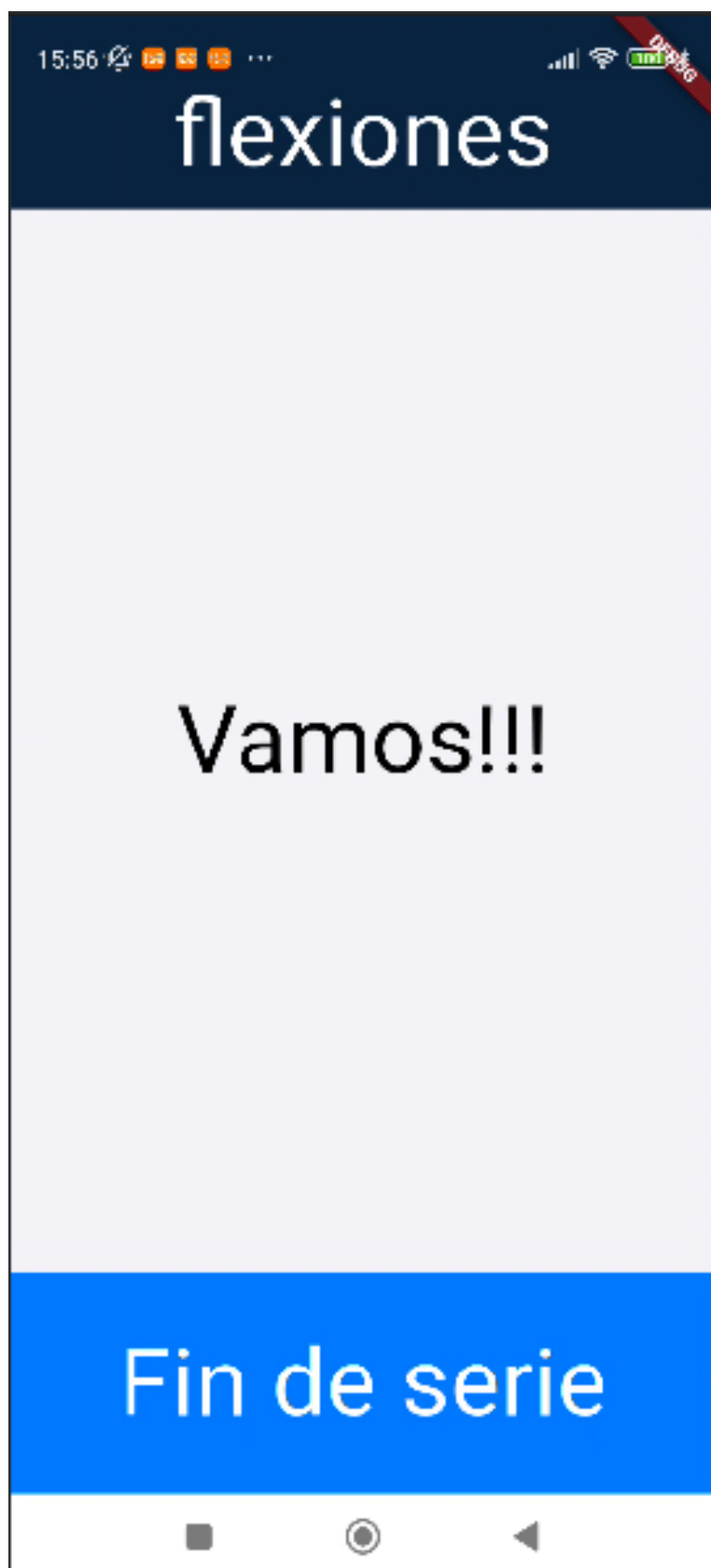


Figura 3.86: Realizando serie

The image shows a mobile application interface with a dark blue header containing the title "Marcas". Below the header is a light gray area with four blue rectangular input fields stacked vertically, labeled "REPETICIONES", "PESO", "TIEMPO", and "DISTANCIA". Below these fields is a text display showing "Descanso: 0 : 0". At the bottom of the screen is a large orange button labeled "Guardar". The status bar at the top shows the time 15:56, battery level, and signal strength. A red diagonal banner in the top right corner says "gratis".

Figura 3.87: Guardando marcas

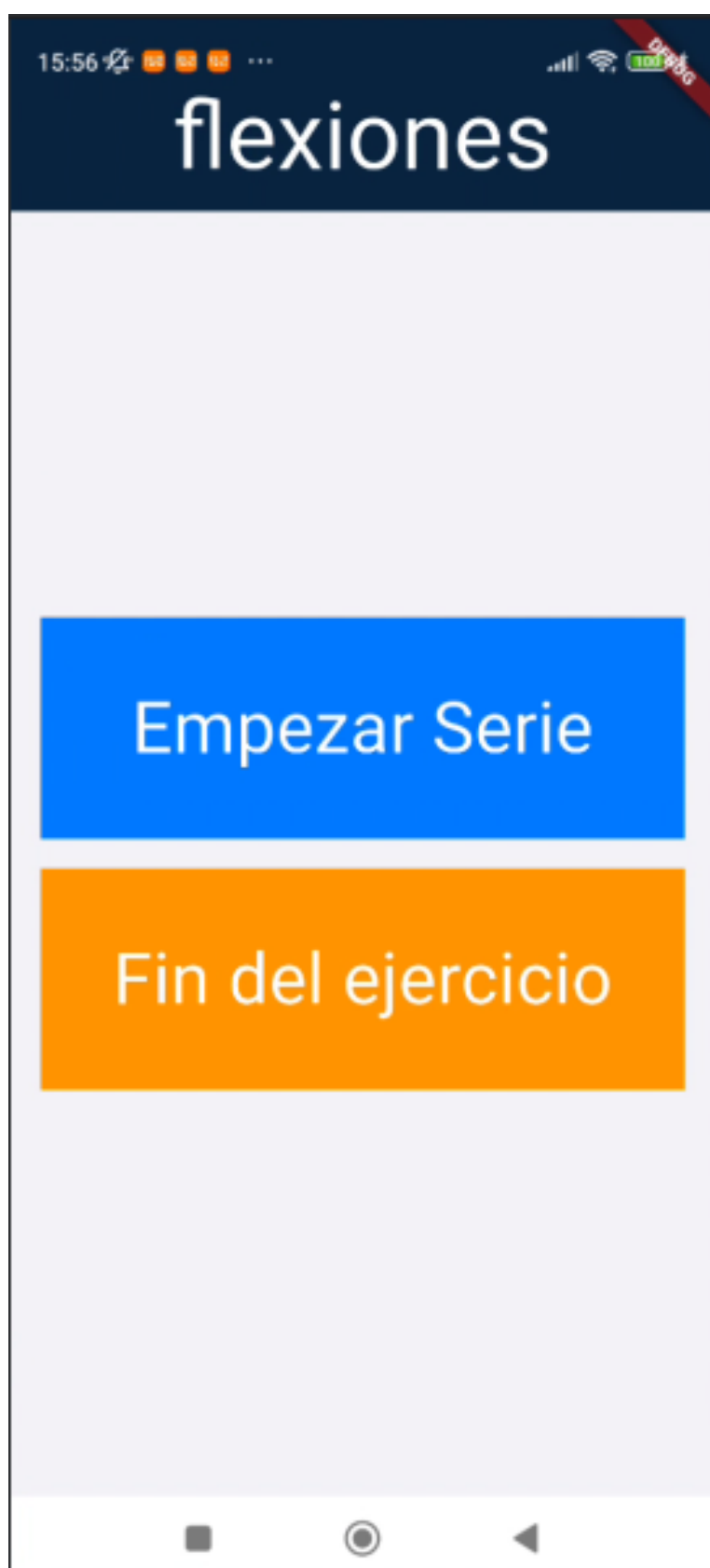


Figura 3.88: Pantalla enter series

Para acaparar esta funcionalidad, se ha implementado una clase aparte encargada del desarrollo de esta parte de la app. Se ha decidido así ya que es una lógica más compleja y es conveniente tenerla separada de la IU.

Esta clase, solo se puede crear de forma util llamando a un método estático de la propia clase (se hace así ya que necesita hacer operaciones asíncronas para obtener los datos necesarios), este método estático me devuelve la instancia con la que voy a trabajar. Se llama al método ejecutar de la instancia creada y sola se encarga de mostrar las pantallas al usuario, guardar los datos y de comparar las metas que se propuso el usuario.

Aquí un diagrama de actividades que explica la tarea de esta clase:

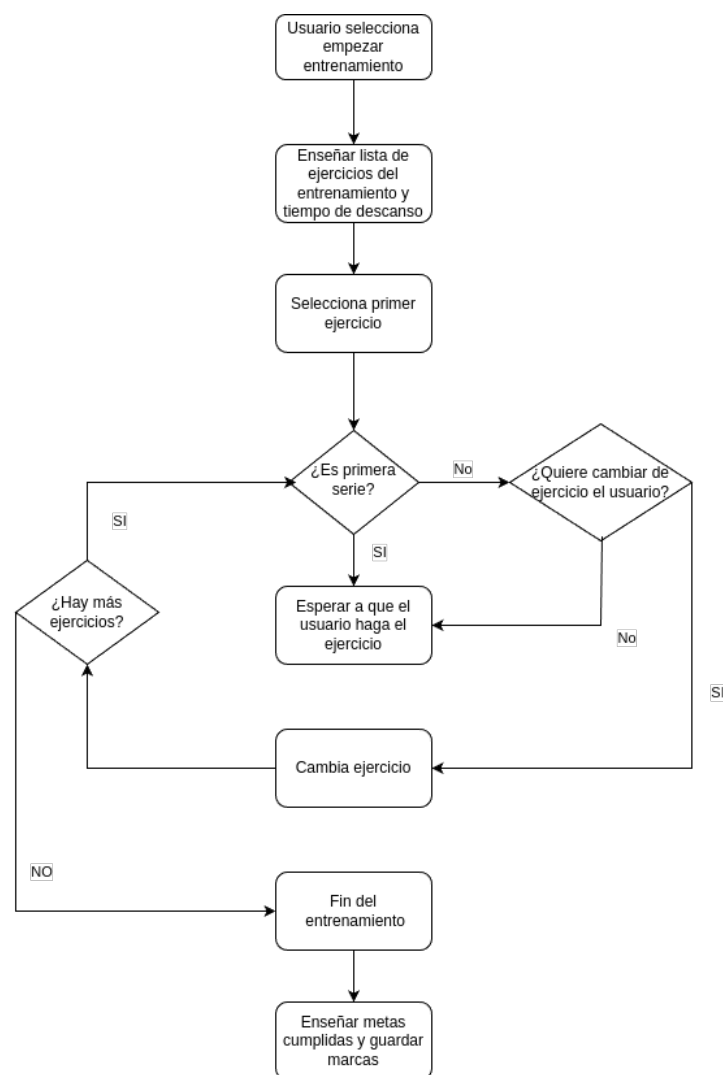


Figura 3.89: Flujo entrenamiento

Se hicieron pruebas simulando el flujo varias veces de distintas formas, comprobando su correcta ejecución y que se guardaban correctamente los datos. Pasó todas las pruebas.

SCRUM-26 Detectar metas cumplidas en los ejercicios despues del entrenamiento
Tareas: 1. Implementar bocetos de IU fig. 3.42 2. Implementar lógica para comparar marcas y metas despues de acabar el entrenamiento en la clase que controla el flujo del entrenamiento
Pruebas de aceptación: <ul style="list-style-type: none">■ si en un ejercicio no se ha cumplido la meta, no se enseña nada■ si en un ejercicio cumplimos una meta, se enseña en que ejercicio se cumplió la meta

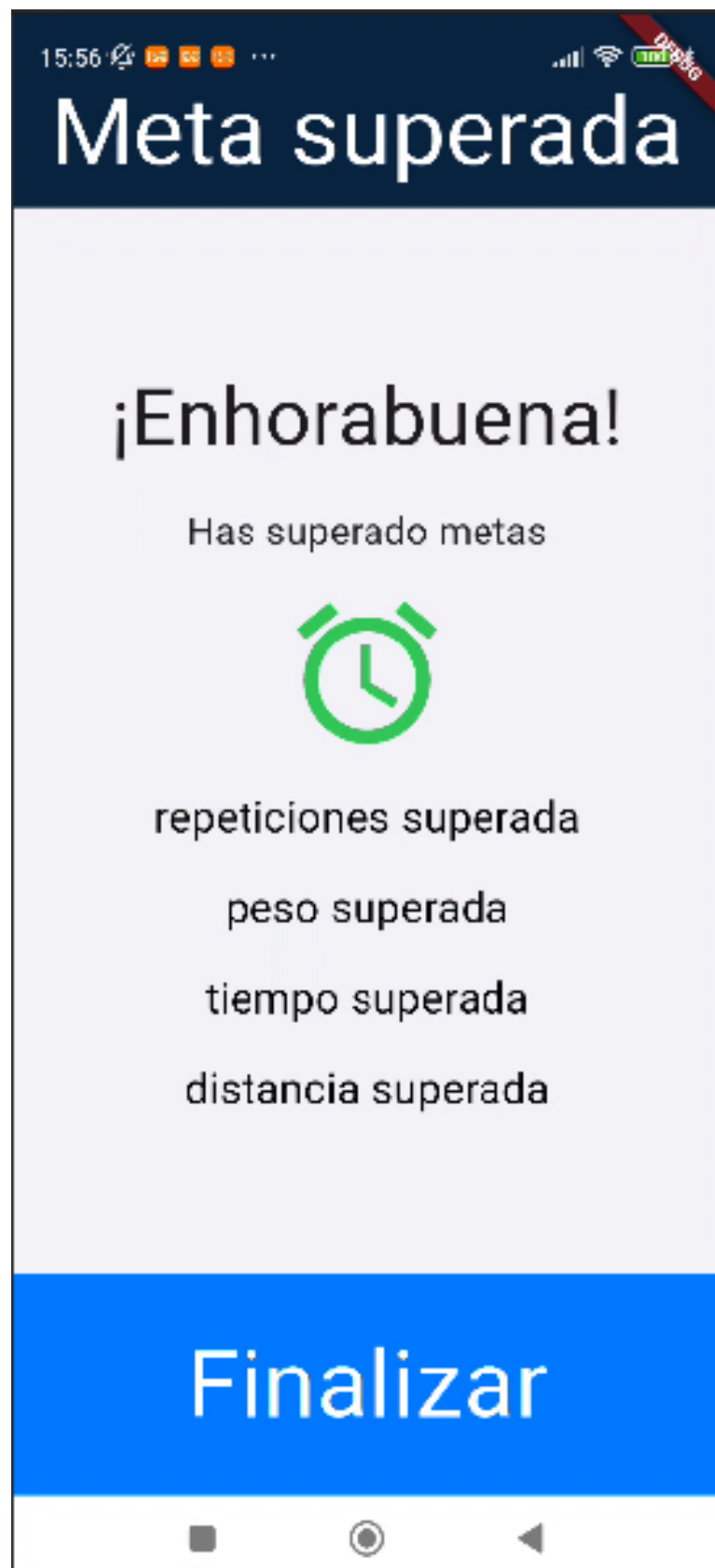


Figura 3.90: Meta superada

Esta funcionalidad es un añadido a la clase FlujoEntrenamiento, cuando se detecta el fin antes de subir las marcas, se ordenan por ejercicio y se comprueba si en algún momento se superó alguna meta y cual parámetro medido fue el superado. Se hicieron pruebas comprobando si de verdad se cumplían las metas cuando saltaba el aviso en pantalla, pasó todas las pruebas.

SCRUM-1 Registrar peso por día
Tareas: <ol style="list-style-type: none">1. Implementar boceto de IU fig. 3.452. Implementar en BD local3. Implementar funciones de inserción4. Implementar funciones de consulta5. Implementar lógica para determinar si un usuario quiere subir o bajar de peso
Pruebas de aceptación: <ul style="list-style-type: none">■ si se introduce algún dato como formato incorrecto, se muestra una alerta por pantalla■ si se introducen los datos en formato correcto, se guarda

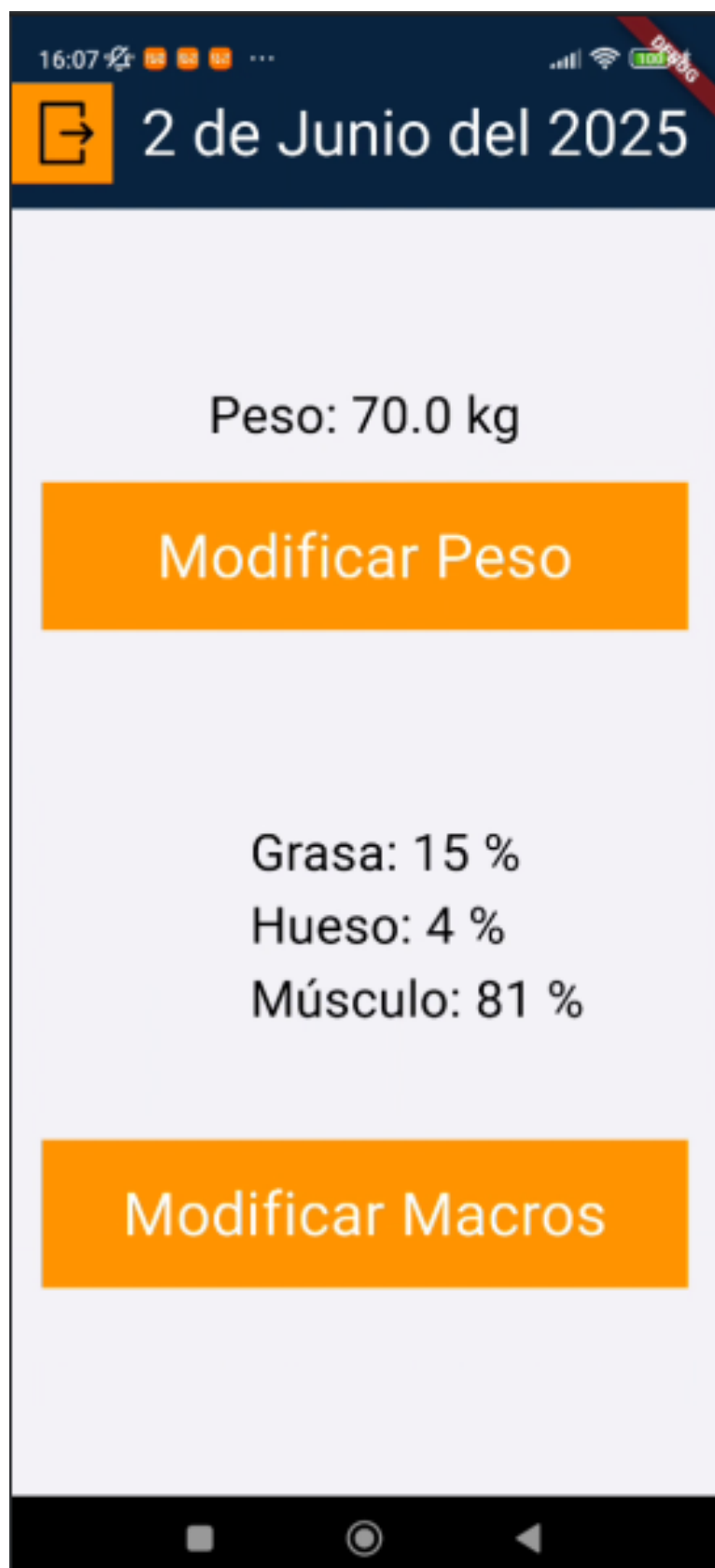


Figura 3.91: Registrar peso por día

Para implementar esta funcionalidad se añadió un acceso en la parte de abajo de la pantalla para cada día del calendario, dando igual si existe algún evento o no, siempre se puede guardar/modificar el peso del usuario siempre que no sea de un día pasado. Se realizaron pruebas comprobando los correctos formatos al introducir el peso y los porcentajes, también se comprobó que se hacía un correcto display de estos datos al acceder a ellos desde el calendario. Pasó todas las pruebas.

En esta iteración hay pocas funcionalidades comparadas con otras, porque el flujo del entrenamiento es una funcionalidad muy grande. A parte se añade esta funcionalidad que también se va a realizar en esta iteración, SCRUM-28 Implementar calendario.

3.9.5.1. Cambios en el backend

Los cambios en el backend están relacionados con las siguientes funcionalidades propuestas en la iteración anterior:

- Cambio 1: Verificar el token del usuario antes de permitir la subida de rutinas, para evitar suplantación.
- Cambio 2: Marcar a los usuarios permanentemente eliminados con una *flag*, para proceder a eliminarlos en los dispositivos.

Cambio 1: la solución implementada es la siguiente, a todas las funciones que puedan ser potencialmente víctimas de una suplantación (vía petición normal de http), se les ha añadido un middleware, una función que se ejecuta antes para verificar el token. Express.js permite implementar esto rápidamente.

Cambio 2: de primeras se pensó en usar una *flag*, pero finalmente para evitar pérdida de rendimiento al momento de que el usuario realice un entrenamiento (que es cuando en un principio se tenía pensado borrar el ejercicio), se optó por modificar la lista de ejercicios de la rutina al momento de borrar el ejercicio. El número de consultas iba a ser el mismo, pero se ahorra memoria, ya que no añadimos una columna más a la tabla de ejercicios.

Otro cambio en el backend que no tiene que ver con lo propuesto con la iteración anterior, la adición de 2 campos a la tabla usuarios del backend, se añadió el campo *int* descargas, siendo este el número total de descargas que posee en sus rutinas el usuario en cuestión, y el campo *fechaCreacion* que es la fecha en la que se creó la cuenta en cuestión. El número total de descargas se usa para que cuando se busquen usuarios para descargar rutinas salgan los que más descargas poseen primero y la fecha de creación para que cuando la app se inicie cree un calendario que vaya desde la fecha de creación hasta 1 mes en adelante de la fecha actual, porque no tiene sentido añadir fechas de antes de crear la cuenta.

3.9.5.2. Estado de la BD local

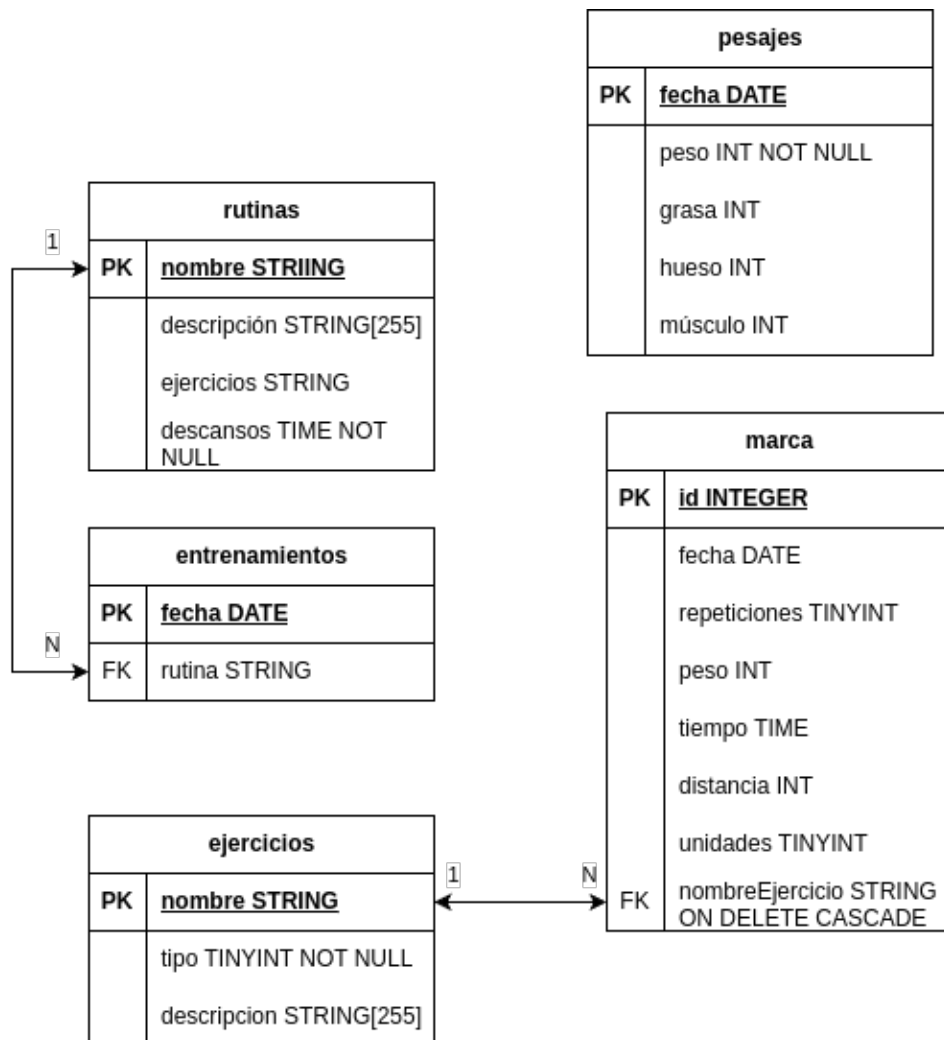


Figura 3.92: BD local iteración 4

3.9.5.3. Estado BD backend

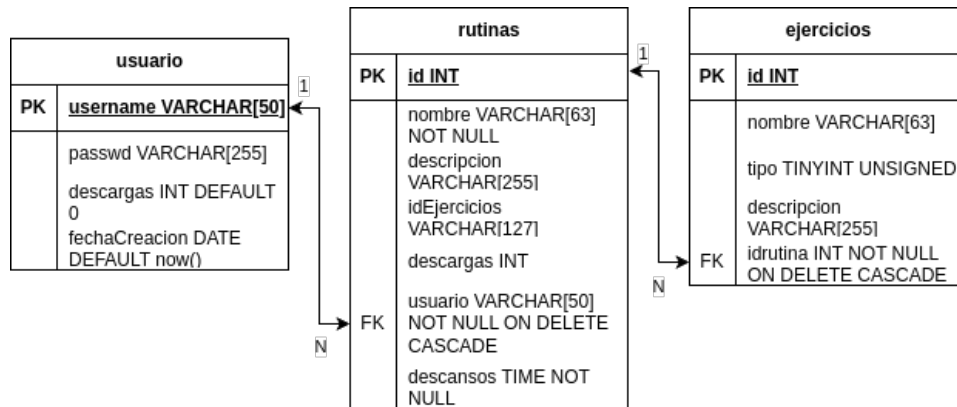


Figura 3.93: BD backend iteración 4

3.9.5.4. Sprint review 4

En esta review se ha propuesto que después de los entrenamientos se les enseñe a los usuarios las metas que tenían establecidas siempre, aunque no la hayan superado, evitando siempre el feedback negativo al usuario. Se tendrá en cuenta para la próxima iteración.

3.9.6. Iteración 5

Al principio de la iteración se dio a notar que faltaba la IU y algunas funciones para visualizar las marcas obtenidas por el usuario, se añadirá a esta iteración. Teniendo en cuenta el tiempo que se posee la IA no se podrá implementar se dará prioridad a la funcionalidad de hacer gráficas usando las marcas obtenidas por el usuario, no se añadió desde un principio porque era necesario poder añadir marcas.

- SCRUM-34: Enseñar las marcas obtenidas en un día
- SCRUM-6: Hacer gráfica en base a las marcas obtenidas

SCRUM-34: Enseñar las marcas obtenidas en un día
Tareas:
1. Implementar las consultas en la BD local
2. Implementar la IU
Pruebas de aceptación:
<ul style="list-style-type: none"> ■ si existen varios ejercicios separarlos en distintas pantallas ■ pulsar en la pantalla para ir pasando series del mismo ejercicio

Se añadió una nueva pantalla MarcasEntrenamientos, en la cual se muestran los resultados del entrenamiento de un usuario en sus distintas series. Pasó todas las pruebas.

SCRUM-6: Hacer gráfica en base a las marcas obtenidas
Tareas:
1. Implementar las consultas necesarias
2. Implementar la IU
Pruebas de aceptación:
<ul style="list-style-type: none"> ■ hacer un display correcto de los distintos datos

Esta funcionalidad dado al poco tiempo que se posee no se implementará